DOI: 10.3745/KIPSTC.2009.16-C.2.245

# Doughnut: 효율적인 지역성 및 캐싱을 사용하는 향상된 P2P Pastry 오버레이 네트워크

김 명 원[†] · 곽 후 근[††] · 정 규 식[†††]


## 요 약

Pastry 오버레이 네트워크는 분산 해쉬 테이블(DHT : Distributed Hash Table)을 사용하는 구조적(Structured) P2P이다. Pastry에서는 노드들 사이의 메시지 수를 줄이기 위해 각각 공간적 지역성과 캐싱을 이용한 Rosary와 LAR이 제안되었다. Rosary는 Inter-Pastry와 Intra-Pastry로 구성된다. Rosary에서 루트 노드는 각 Intra-Pastry를 대표하는 노드가 할당되고 Inter-Pastry와 Intra-Pastry 라우팅을 책임진다. 이러한 구조로 인해 Rosary는 다음과 같은 단점을 가진다. 첫째는 루트 노드의 실패 시 고장 방지 능력(Fault Tolerance)에 약하다는 점이고, 둘째는 루트 노드를 사용하기 때문에 라우팅 홉 카운트가 기존 Structured P2P에 비해 증가한다는 점이다. 마지막으로 셋째는 통신 부하가 특정 지역에 집중한다는 점이다. LAR의 경우 캐싱이 Intra-Pastry내의 노드들 사이에 골고루 분포되지 않고 Intra-Pastry내의 특정 노드들에 의해서만 사용되어지는 단점을 가진다.

본 논문에서는 Rosary와 LAR의 문제점을 해결한 Doughnut이라 불리는 개선된 Pastry를 제안한다. Doughnut은 지역적 특성에 따라 노드들을 구분한 Inter-Pastry와 Intra-Pastry로 구성되고, 모든 노드들은 Inter-Pastry와 Intra-Pastry 라우팅을 책임진다. 이것은 모든 노드들이 기존의 루트 노드의 역할을 수행함을 의미한다. 이러한 방법은 고장 방지 능력이 감소하는 문제, 라우팅 홉 카운트가 증가하는 문제 및 통신 부하가 균일하게 분포하지 않는 문제를 해결한다. 또한 Doughnut은 지역적으로(Intra-Pastry) 캐쉬의 균일한 분포를 보장하고, 지역안의 캐쉬 콘텐츠는 다른 지역에서도 사용될 수 있기 때문에 효율적으로 캐쉬를 사용할 수 있다. 제안된 알고리즘은 시뮬레이터를 통해 구현되었고, 실험 결과는 기존 방법에 비해 제안된 방법이 효과적임을 보여준다.


**키워드 : 오버레이 네트워크, 지역성, 캐싱, Pastry, Rosary, LAR, Doughnut**

# Doughnut: An improved P2P Pastry Overlay Network with Efficient Locality and Caching

Myungwon Kim[†] · Hukeun Kwak[††] · Kyusik Chung[†††]


## ABSTRACT

Pastry overlay network is one of structured P2Ps using DHT(Distributed Hash Table). To reduce the number of messages among nodes, Rosary and LAR have been proposed by exploiting spatial locality and caching, respectively, in the Pastry. Rosary consists of Inter-Pastry and Intra-Pastry. A root node is assigned as a representative in each Intra-Pastry and it has the responsibility of Inter-Pastry and Intra-Pastry routing. Therefore, Rosary has several disadvantages; 1) low fault tolerance in case of root node failure 2) routing hop count increases because of the use of root nodes compared to the existing structured P2Ps, and 3) the communication load is concentrated in some specific areas. LAR has inefficient problems in that caching is not distributed among nodes in Intra-Pastry and caching is used by only nodes in the Intra-Pastry.

In this paper, we propose an improved Pastry called Doughnut to overcome the above problems of Rosary and LAR. By dividing nodes with the local characteristics, the Doughnut consists of Inter-Pastry and Intra-Pastry, and all nodes have the responsibility of Inter-Pastry and Intra-Pastry routing. This results in that all nodes perform the role of the existing root node. This solves the problems of the reducing of fault-tolerance, the increasing of routing hop count, and the not-distributed communication load. Also Doughnut can use cache effectively because it guarantees the even cache distribution in local(Intra-Pastry) and the cache contents in local can be used in the other local. The proposed algorithm is implemented using simulator and the experimental results show the effectiveness of the proposed method compared to the existing method.


**Keywords : Overlay Network, Locality, Caching, Pastry, Rosary, LAR, Doughnut**

※ 본 연구는 숭실대학교 교내 연구비 지원으로 이루어졌음.
† 정 회 원 : 숭실대학교 정보통신전자공학부 석사과정
†† 정 회 원 : 숭실대학교 정보통신전자공학과 postdoc(교신저자)
††† 정 회 원 : 숭실대학교 정보통신전자공학부 교수
논문접수 : 2008년 8월 26일
수 정 일 : 1차 2008년 12월 5일
심사완료 : 2008년 12월 29일

## 1. 서 론

P2P 시스템은 네트워크 환경에서 집중화된 서비스 개념 없이 분산된 자원의 공유를 목적으로 동등한 자격을 가진 자율적(autonomous) 객체(피어)로 이루어진 자율 구성 시스템으로 정의된다[1]. 이는 기존의 인터넷에서 사용되던 서버-클라이언트 개념의 단 방향 특성이 정보를 유기적으로 교환하는 휴먼 커뮤니케이션의 상황에 적합하지 않기에 고안된 양방향 커뮤니케이션 모델이다[2]. P2P 시스템은 파일 공유를 위한 Napster[3]의 소개로 알려지기 시작하여 분산 컴퓨팅(SETI@home)[4], 인터넷 전화(Skype)[5], IPTV(Zoost)[6] 등에 성공적으로 적용되었으며, 현재 가장 관심 있는 인터넷상 새로운 통신방식으로 떠오르고 있다.

P2P 시스템은 크게 비구조적(Unstructured) 방식과 구조적(Structured) 방식 두 가지로 분류할 수 있다. 비구조적 P2P 시스템의 종류로는 플러딩(Flooding)을 기반으로 한 순수(Pure) P2P 방식(Gnutella[7])과 서버를 기반으로 한 중앙집중적(Centralized) P2P (Napster[3])방식, 그리고 두 가지 방식의 혼합형인 혼합(Hybrid) P2P 방식(JXTA[8])이 존재 한다. 그러나 순수 P2P 방식은 플러딩에 따른 네트워크 트래픽을 발생 시키며 중앙집중적 P2P 방식이나 혼합 P2P 방식은 전체 네트워크에 대한 정보들이 몇몇 노드들에 의해 관리되기 때문에 확장성 및 고장 방지 능력에서 단점을 갖는다.

구조적 P2P 시스템은 각각의 노드가 전체 네트워크가 아닌 부분적인 네트워크 정보를 유지 및 관리하게 함으로써, 비구조적 P2P 시스템의 단점을 보완한 방법이다. 해쉬 알고리즘(SHA-1등)을 사용하여 콘텐츠와 피어 정보들을 공통의 단일 주소 공간으로 매핑하는 콘텐츠-어드레싱 기반 데이터 저장 기법을 사용하고, DHT(Distributed Hash Table) 알고리즘을 이용하여 오버레이 네트워크 위에 존재하는 다양한 콘텐츠들을 적절히 분산 시킨다 그리고 룩업 지연(Lookup Latency)을 최소화 하여 분산된 콘텐츠를 찾을 수 있는 방법을 제공한다. 이러한 특성은 구조적 P2P 환경에서 노드의 참여 및 이탈(Join/Leave)을 자동화하여 관리 부담이 적고, 각종 노드 실패의 경우에도 자동 복구가 가능케 한다. 구조적 P2P 시스템의 분류는 DHT 알고리즘에 따라 구분될 수 있으며 2진법을 사용한 CHORD[9], 2b진법을 사용한 Pastry[10], d-차원 데카르트 좌표계에 기반을 둔 CAN[11] 등으로 나눠진다. 각각의 기술들은 확장성 있고 장애에 강한 분산 해쉬 테이블을 이용하는 공통점을 갖는 반면 네트워크 근접성, 라우팅 테이블 크기, 룩업 카운트 등에서 차이점을 보인다.

그러나 키(Key)값을 통한 라우팅은 실제 물리적으로 근접한 노드들이 전송 대상에서 제외 될 수 있는 단점을 갖는다. 이는 DHT를 이용한 라우팅은 최적의 전송 경로를 갖지 못하는 상황을 발생 시킨다. Doughnut은 이러한 문제를 극복하기 위해 Pastry 오버레이 네트워크상에서 모든 노드가 공평한 역할을 분담하는 지역성(Locality) 개념을 사용한다. Doughnut은 Pastry 뿐만 아니라 Tapestry, Kademlia와 같은 Plaxton 알고리즘을 사용하는 구조적 P2P 알고리즘에 적용할 수 있으나 본 논문에서는 구조적 특성을 잘 표현할 수 있는 Pastry 알고리즘을 모델로 사용하였다.

본 논문의 구성은 다음과 같다. 2장에서는 연구의 기반이 되는 Pastry 알고리즘과 Pastry 알고리즘에서 지역성과 캐싱(Caching)에 대한 기존 연구를 설명한다. 3장에서는 지역성과 캐싱을 위한 효율적인 알고리즘을 소개한다. 4장에서는 제안한 알고리즘에 대한 시뮬레이션 모델 결과를 기술하였고 마지막으로 5장에서 결론을 맺는다.

## 2. 기존연구

### 2.1 Pastry

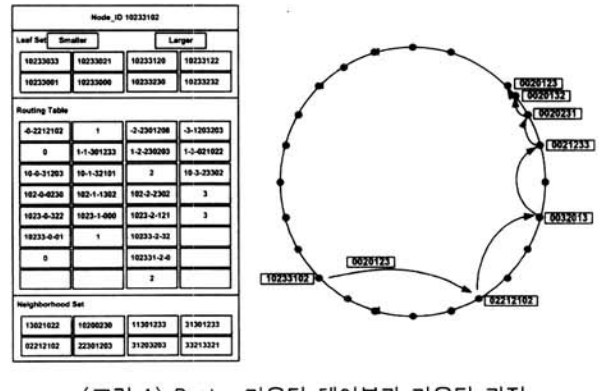
Pastry[10]는 DHT(Distributed Hash Table)를 사용하는 대표적인 구조적 P2P 방식의 오버레이 네트워크 알고리즘으로써 확장성이 있고 장애에 강한 특성을 갖는다. 주소공간의 크기가 128 bit로 고정되어 있고 2b(일반적으로 b=4) 진법이라는 임의의 진법을 사용한다. Pastry 네트워크를 구성하고 있는 모든 노드들은 IP 등의 정보를 바탕으로 0 ~ 2128-1 사이의 고유의 아이디(ID)값을 가지며, 오버레이 네트워크상의 콘텐츠는 해쉬 함수를 이용하여 128 bit의 키값을 갖게 되고 해당 키 값과 동일하거나 가장 근접한 아이디를 갖는 노드들에게 배포되어진다. 콘텐츠의 검색 방법은 콘텐츠의 이름 정보로부터 키 값을 구하고, 각각의 노드들이 갖고 있는 라우팅 정보를 근거로 하여 키 값과 동일하거나 유사한 아이디를 갖는 노드에게 전해진다.

#### 2.1.1 라우팅 테이블

Pastry를 구성하는 각각의 노드들은 라우팅 정보를 위해 Leaf Set, 라우팅 테이블(Routing Table), Neighborhood Set의 3개의 테이블을 유지 및 관리한다. 각 테이블의 용도는 다음과 같으며 (그림 1)의 왼쪽 그림에 보인다.

Leaf Set : 주소 공간 내에서 현재 노드에 가장 근접한 L(일반적으로 L = 16 혹은 32)개의 노드들의 정보를 저장한다. 노드 아이디보다 큰 |L|/2개의 노드들과 노드 아이디보다 작은 |L|/2개의 노드들의 정보를 기억한다.

라우팅 테이블 : 라우팅 테이블은 128/2b 행과 2b 열로




(그림 1) Pastry 라우팅 테이블과 라우팅 과정

구성된다. 각 행은 현재노드와 다른 노드들 사이의 공통 프리픽스(Prefix)의 길이가 짧은 원격 노드들을 가장 먼저 배치하고, 공통 프리픽스가 긴 것들을 테이블의 아래쪽에 배치한다.

Neighborhood Set : 현재 노드와 네트워크상으로 가장 근접한 L개의 노드들의 정보를 저장한다.

#### 2.1.2 라우팅 알고리즘

Pastry에서의 라우팅은 라우팅 테이블의 3가지 요소 중 가장 먼저 Leaf Set를 찾아서 네트워크상에서 가장 근접한 노드를 가져온다. 그리고 Leaf Set에 존재하지 않을 때 라우팅 테이블을 찾아보고 가장 긴 공통 프리픽스를 지닌 노드로 쿼리(Query)를 전달하여 라우팅을 행하게 된다. 만약에 라우팅 테이블에서도 목표 노드를 찾을 수 있는 링크가 존재하지 않는다면, 현재 노드가 가지고 있는 정보 중 목표 노드의 주소에 가장 근접한 주소를 가진 노드에 쿼리를 재전송하여 라우팅을 하게 된다. 이러한 라우팅 방법을 통하여 Pastry에서는 log2bN의 라우팅 성능을 가질 수 있다. <표 1>은 Pastry의 라우팅 알고리즘을 의사코드화한 것이며, (그림 2)의 오른쪽 그림은 Pastry노드에서 메시지 라우팅을 보여준다.

#### 2.1.3 자기 조직화(Self-Organization)

새로운 노드가 Pastry 오버레이 네트워크에 조인할 때 노드 아이디 X를 갖는 새 노드는 기존의 Pastry 노드 A에 반드시 접촉해야 한다. 그 후 A는 키 값으로 X의 노드 아이디를 사용하여 메시지를 라우팅 한다. 새로운 노드 X는 노드 A부터 X까지의 경로에서 만난 노드로부터 라우팅 테이블의 n번째 행을 얻는다. 이렇게 하여 얻어진 라우팅 테이블은 주기적인 라우팅 테이블 유지 작업을 통하여 유지 관리 된다.

Pastry에서 라우팅 테이블에 근거한 라우팅 방법은 콘텐츠를 소유한 노드가 물리적으로 근접 하더라도 전송 대상에서 제외됨으로써 최적의 전송경로를 찾지 못하는 문제를 발생 시킨다. 이러한 구조적 P2P의 구조적 문제는 현재 학계에서 논의되는 분야이며 효율적인 P2P 환경을 위해 해결해야 할 숙제이다.

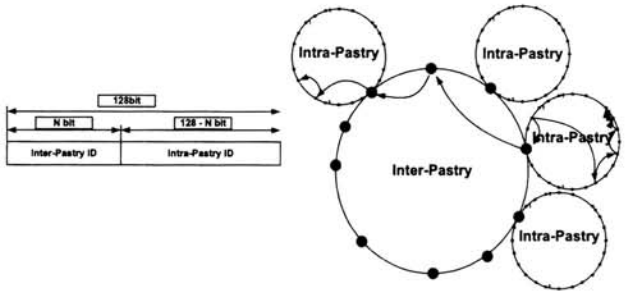
### 2.2 지역성(Rosary)

지역성는 라우팅 테이블에 물리적으로 근접한 노드들을 포함 하도록 하는 방법이다. 남궁정일[12]은 Rosary에서 Inter-Pastry와 Intra-Pastry로 구분되는 개념을 소개하였다. Rosary는 지역적으로 근접한 노드를 Intra-Pastry로 구성하고 Intra-Pastry들의 루트(Root) 노드, 즉 Intra-Pastry를 대표하는 노드들만을 연결하여 Inter-Pastry를 구성한다. (그림 2)의 왼쪽 그림은 Inter-Pastry 아이디와 Intra-Pastry 아이디 영역으로 나뉘는 Rosary의 노드 아이디 구조를 나타내며, (그림 2)의 오른쪽 그림은 Rosary의 구조와 라우팅 과정을 보여준다. Rosary에서 각각의 노드는 콘텐츠를 지역 내에서 찾도록 시도하며 이때 Intra-Pastry 아이디가 이용된다. 지역 내에서 콘텐츠의 부재 시 메시지는 루트 노드로 전송되어 Inter-Pastry 라우팅 과정을 수행하게 된다. 루트 노드는 Inter-Pastry 라우팅을 거쳐 해당 콘텐츠의 키 값과 근접한 아이디를 갖는 노드가 속한 지역의 루트 노드에게 메시지를 전송하게 되며 다시 Intra-Pastry 라우팅을 거쳐 최종적으로 키 값과 근접한 아이디를 갖는 노드에게 전송되어진다. 루트 노드는 Intra-Pastry 내의 노드들 중 높은 처리 능력을 가지며 오버레이 네트워크에 오랫동안 연결된 노

〈표 1〉 Pastry 라우팅 알고리즘 (의사코드)

$R_l^i$ : 열 i 에서 라우팅 테이블 R의 요소, $0 \le i \le 2^b$이며 행 l , $0 \le l \le 128/b$
$L^i$ : Leaf Set L 에서 i 번째 근접한 노드 아이디, $-|L|/2 \le i \le |L|/2$
$D_l$ : 키 D에서 l 위치의 값
shl(A,B) : A와 B 사이 프리픽스 공유 길이

```
if (L_|L|/2 ≤ D ≤ L_|L|/2) {
    // D is within range of our leaf set
    Forward to L_i, such that |D-L_i| is minimal;
} else {
    // 라우팅 테이블 이용
    Let l = shl(D,A);
    if (R_l^Dl ≠ null) {
        Forward to R_l^Dl;
    } else {
        // Rare case
        Forward to T∈L∪R∪M, such that shl(T,D) ≥ l, |T-D| < |A-D|;
    }
}
```




(그림 2) Rosary 아이디 구조와 라우팅 알고리즘

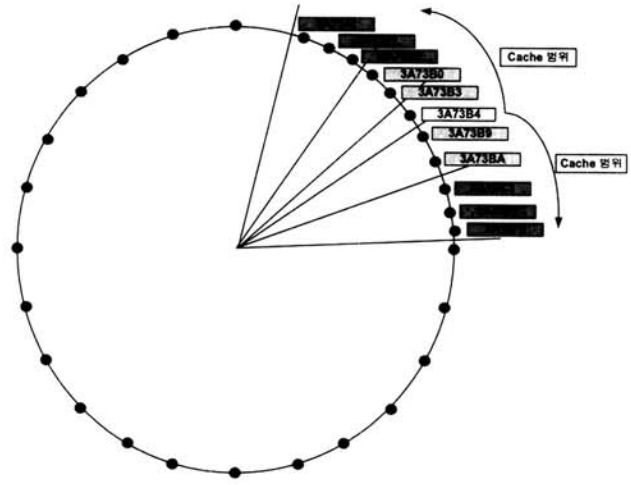
드가 맞는다. 루트 노드가 오버레이 네트워크를 떠날 경우 남아있는 Intra-Pastry 노드들 중 추천된 임의의 노드가 루트 노드의 역할을 수행하게 된다.

그러나 Rosary 구조에서 루트 노드로서 동작하는 노드는 높은 네트워크 부하와 Inter-Pastry에 대한 추가적인 정보의 유지가 요구되기에 공평성에서 문제가 될 수 있으며 루트 노드의 실패는 지역 전체에 영향을 미치므로 기존 구조적 P2P의 장점인 고장 방지 능력 측면이 약화된다는 점에서 큰 단점을 갖는다. 또한 새로운 루트 노드가 선출될 경우 새로운 루트 노드에 대한 정보는 Intra-Pastry 내의 모든 노드들에게 알려져야 하기에 새로운 네트워크 부하를 발생 시킬 수 있으며 지역 내에 캐쉬된 콘텐츠는 지역 내의 노드들에게만 사용될 수 있는 캐쉬 비효율성 문제를 갖는다. 라우팅 관점에서는 Intra-Pastry 라우팅, Inter-Pastry 라우팅 그리고 다시 Intra-Pastry 라우팅 과정을 필요로 함으로써 기존 Pastry 알고리즘 보다 홉 카운트(Hop Count)가 많이 증가 한다.

### 2.3 캐싱(LAR)

오버레이 네트워크상에는 다양한 종류의 콘텐츠가 존재하며 각각의 콘텐츠에 대한 사용자의 요청 또한 다르다. 인기 있는 콘텐츠를 갖는 노드와 그렇지 않는 노드는 콘텐츠 요청의 처리량이 다르기에 노드 사이의 공평성을 제공하지 못하며, 인기 있는 콘텐츠 전송을 한 노드가 담당하는 방법은 오버레이 네트워크의 콘텐츠 전송의 품질(Quality) 또한 떨어뜨린다. 이러한 문제를 해결하기 위한 방법으로 가상 서버(Virtual Server)를 이용하여 콘텐츠를 부하가 적은 노드에게 넘겨주는 방법[13]과 인기 있는 콘텐츠를 오버레이 네트워크 위의 여러 노드에 캐싱을 하는 방법이 있다[14-16]. 가상 서버 방법은 노드의 공평성을 제공 하지만 결국 콘텐츠를 제공하는 노드는 제한적이기에 콘텐츠 전송의 품질은 제한적이다. 콘텐츠 캐싱 방식은 노드의 부하를 분배 시킬 뿐만 아니라 콘텐츠를 제공하는 노드가 많아짐으로써 품질 또한 보장할 수 있다.

Shen Haiying[16]은 프리픽스에 따른 콘텐츠 캐쉬 방법인 LAR(Locality-Aware Randomized)을 제안 하였다. 콘텐츠의 인기도에 의하여 공통 프리픽스 길이를 줄여가며 캐쉬되는 범위를 넓히는 LAR은 동적인 파일의 캐쉬 방법일 뿐만 아니라 홉 카운트 또한 줄일 수 있는 방법으로써 (그림 3)에 보인다. 그림에서 "3A73B4" 는 자신이 소유한 콘텐츠의 요청이 많을 경우 프리픽스에 따라 분배를 하게 된다. 자신의 아이디와 "3A73B" 가 동일한 노드에게 콘텐츠를 캐싱하며, 이후 "3A73", "3A7" 등 프리픽스 범위를 넓히며 인기도에 따라 콘텐츠 캐싱 범위를 확장 시킨다. 그러나 LAR은 지역성 개념을 고려하지 않은 알고리즘으로써 지역적으로 구분된 Overlay Network에 적용할 경우 지역적으로 불균등한 콘텐츠 캐쉬를 야기할 수 있다. 몇몇 특정 지역에만 집중된 콘텐츠 캐쉬는 다른 지역의 노드가 이를 이용할 때 지역 간의 메시지를 야기 시키기에 비용이 크기에 모든 지역에 균등히 콘텐츠 캐쉬를 배포할 수 있는 특성이 필요하다.




(그림 3) LAR(Locality-Aware Randomized)

### 2.4 접근방식

본 절에서는 기존 지역성 알고리즘과 캐싱 알고리즘이 갖는 단점을 정리 및 본 논문의 접근 방식을 간략하게 설명하고 자세한 내용은 3장에서 기술 한다.

(1) 기존 알고리즘(지역성 및 캐싱)의 문제점

Pastry 알고리즘에서 지역성 고려하기 위한 Rosary는 지역을 대표하는 루트 노드를 생성하기에 루트 노드의 실패는 지역 내 모든 노드의 실패를 야기한다. 이러한 특성은 기존 Pastry의 장점인 고장 방지 능력을 약화 시킬 뿐만 아니라, 루트 노드는 다른 노드에 비해 처리량이 증가함으로써 노드 간 불균등한 부하를 갖게 되는 원인이 된다. 또한 지역 내 모든 노드들은 지역을 대표하는 루트 노드를 인지하고 있어야 하기 때문에 이로 인한 추가적인 네트워크 부하가 발생하며, 기존 Pastry 알고리즘보다 라우팅 홉수가 증가하는 단점을 갖는다.

콘텐츠 캐쉬 측면에서 Rosary의 캐쉬 방법은 모든 콘텐츠 캐쉬가 모든 지역에 분배 되어야 한다는 단점을 갖으며, 지역 내 캐쉬된 콘텐츠는 해당 지역의 노드들만 접근할 수 있다는 비효율성 또한 문제가 된다. 공통 프리픽스에 따른 콘텐츠 캐쉬 방법(LAR)은 콘텐츠 캐쉬와 라우팅 홉수를 줄이기 위한 효과적인 방법이지만 불규칙적인 노드들의 아이

디는 이를 지역성에 적용하기 어렵게 하는 단점을 갖는다. 단순 프리픽스에 의한 캐쉬는 몇몇 지역에만 집중 될 수 있기에 지역적인 균일성을 갖기 어려우며 이는 지역 간 메시지를 야기 시키기에 이를 해결할 수 있는 방법이 필요하다.

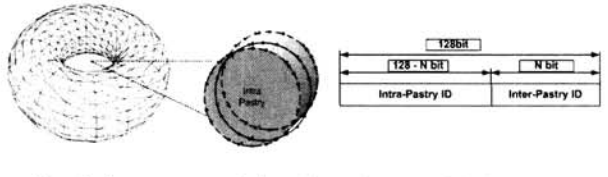
(2) 본 논문의 접근 방식

본 논문에서는 Doughnut 형태의 오버레이 네트워크 구조를 제안한다. Doughnut은 기존의 Rosary에서와 같이 물리적 근접성을 바탕으로 지역을 구분한다. 그러나 지역을 대표하는 루트 노드 대신 각각의 노드들이 루트 노드의 역할, 즉 다른 지역으로 라우팅 할 수 있는 역할을 갖는다. 이러한 특성은 한 노드의 실패 시 다른 노드를 이용하여 Inter-Pastry 라우팅을 할 수 있기에 Pastry 알고리즘의 고장 방지 능력을 유지하며, 모든 노드가 루트 노드로 동작하기에 불균등한 부하문제 또한 해결 할 수 있다. 또한 콘텐츠 캐싱 측면에서 Doughnut은 하위 N 비트를 Inter-Pastry 아이디로 사용함으로써 하위 비트의 아이디 균일성을 보장할 수 있다. 이는 프리픽스에 따른 배포 시 지역적으로 균등한 캐싱를 보장하며 한 지역의 캐싱 콘텐츠는 다른 지역에서 사용할 수 있도록 돕기에 효율적인 콘텐츠 캐싱이 가능케 한다.

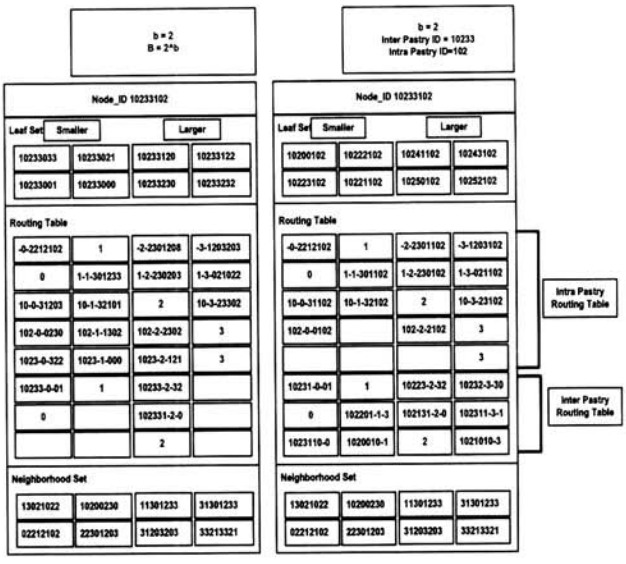
## 3. 제안된 방법

### 3.1 전체 구조

제안된 방식에서는 물리적 근접성을 바탕으로 지역을 구분한다. 물리적 근접성을 얻는 방법으로는 Proximity 측정 방법이나 혹은 ISP(Internet Service Provider)가 제공하는 정보로부터 구분할 수 있다. Doughnut의 전체적인 구성도는 (그림 4)의 왼쪽에 보인다. 각각의 노드들은 지역성으로 구분된 Intra-Pastry를 구성하며, 다른 지역에서 자신의 Intra-Pastry 아이디와 같거나 가장 근접한 Intra-Pastry 아이디를 갖는 노드들과 함께 Inter-Pastry를 구성한다. Doughnut에서 노드 아이디는 (그림 4)의 오른쪽과 같이 구성된다. 노드 아이디의 하위 N 비트는 Inter-Pastry 아이디로, 상위 128-N 비트는 Intra-Pastry 아이디로 이용되며 IP 등의 정보로부터 Intra-Pastry 아이디, 초기에 얻은 지역 아이디를 Inter-Pastry 아이디로 사용한다. 이는 라우팅 테이블을 이용한 라우팅 시 Intra-Pastry, Inter-Pastry 순으로 라우팅이 이루어지도록 함으로써 지역 내에서 콘텐츠 아이디와 근접한 아이디를 갖는 노드를 먼저 찾고 이 후 지역 밖에서 근접한 노드 찾기 위함이다. 이러한 특성은 각 지역에 콘텐츠 아이디와 근접한 아이디를 갖는 노드에 콘텐츠




(그림 4) Doughnut 전체 구성도 및 노드 아이디 구조




(그림 5) Pastry와 Doughnut의 라우팅 테이블

캐싱을 할 경우 효율적인 지역적 서비스를 가능케 한다.

### 3.2 라우팅 테이블

Doughnut의 라우팅 테이블은 (그림 5)의 오른쪽에 보이며, 기존 Pastry의 라우팅 테이블은 (그림 5)의 왼쪽에 나타내었다. 제안하는 방법의 라우팅 테이블은 기존 Pastry와 같이 Leaf Set, Neighbour Set, 라우팅 테이블로 구성되며 추가적인 데이터공간은 필요하지 않다. 각각의 테이블에 관한 정의는 다음과 같다.

Leaf Set : Intra-Pastry 내에서 현재 노드에 가장 근접한 L개의 노드들의 정보를 저장한다.

라우팅 테이블 : $128/2^b$행으로 구성되어진 라우팅 테이블에서 상위 $(128-n)/2^b$행은 Intra-Pastry 라우팅에 사용되며 하위 $n/2^b$행은 Inter-Pastrty에서 사용된다. Intra-Pastry 라우팅 테이블은 기존 Pastry 라우팅 테이블과 같지만 오직 같은 지역 내 노드들로만 구성된다. Inter-Pastry 라우팅 테이블은 지역 아이디 만을 사용하여 공통 프리픽스가 적은 지역을 위에, 많은 지역을 아래에 두며 다른 지역에서 자신의 Intra-Pastry 아이디와 같거나 가장 근접한 아이디를 갖는 노드로 구성된다.

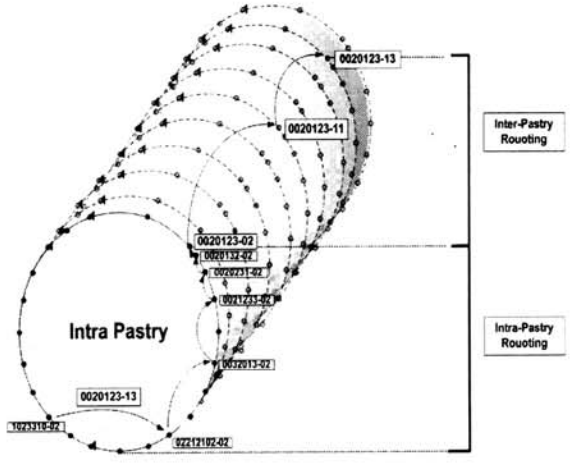
Neighborhood Set : Intra-Pastry내에서 현재 노드와 네트워크상으로 가장 근접한 L개의 노드들의 정보를 기억한다.

### 3.3 라우팅 알고리즘

Doughnut의 라우팅은 기존 Pastry 알고리즘과 같이 먼저 Leaf Set, 라우팅 테이블 순으로 검사되며 Leaf_Set, 라우팅 테이블에서 결정을 내릴 수 없을 경우 갖고 있는 모든 데이터 중 가장 근접한 노드에게 전송하게 된다. Leaf Set 검색의 경우 키 값의 상위 128-N 비트와 Leaf Set 데이터의 Intra-Pastry 아이디값만으로 이루어지며, 이러한 특성은 지역 내에서 키 값의 상위 128-N 비트의 값에 가장 근접한 노드에게 메시지가 전송되게 된다. 만약 노드가 자신이 지

〈표 2〉 Doughnut 라우팅 알고리즘 (의사 코드)

```
Intra-R_l^i : 열 i 에서 라우팅 테이블 R의 요소, 0 ≤ i ≤ 2^b이며 행 l , 0 ≤ l ≤ (128-N)/b
Inter-R_l^i : 열 i 에서 라우팅 테이블 R의 요소, 0 ≤ i ≤ 2^b이며 행 l , (128-N)/b ≤ l ≤ 128/b
L^i : Leaf Set L 에서 i 번째 근접한 노드 아이디, -|L|/2 ≤ i ≤ |L|/2
D_l : 키 D에서 l 위치의 값
shl_intra(A,B) : Intra-Pastry 아이디 에서 A 와 B 사이 프리픽스 공유 길이
shl_inter(A,B) : Inter-Pastry 아이디 에서 A 와 B 사이 프리픽스 공유 길이
if (File_List()) {
    // File is in disk
    Find in File_List();
}
if (L_-|L|/2 ≤ D ≤ L_|L|/2) {
    // D is within range of our leaf set
    // 자신의 Intra-Pastry 아이디가 콘텐츠의 Intra-Pastry 아이디와 가장 근접한 노드이면 Intra-Pastry 라우
    // 팅에서 Inter-Pastry라우팅으로 옮긴다.
    if (My Intra-Pastry 아이디 is the Nearest) goto Inter-Pastry routing;
    // 자신의 Intra-Pastry 아이디가 콘텐츠의 Intra-Pastry 아이디와 가장 근접한 노드가 아니면 Leaf Set 테
    // 이블로부터 가장 근접한 노드를 찾는다.
    Forward to L_i, such that |D-L_i| is minimal;
} else {
    // Use the 라우팅 테이블
    Let l = shl_intra(D,A);
    if(Intra-R_l^Dl ≠ null) {
        Forward to Intra-R_l^Dl;
    }else{
        // Rare case
        Forward to T∈L∪R∪M, such that shl(T,D) ≥ l, |T-D| < |A-D|;
    }
}
Inter-Pastry:
// Inter-Pastry 라우팅 테이블 사용
Let l = shl_inter(D,A);
if (Inter-R_l^Dl ≠ null) {
    Forward to Inter-R_l^Dl;
}
```

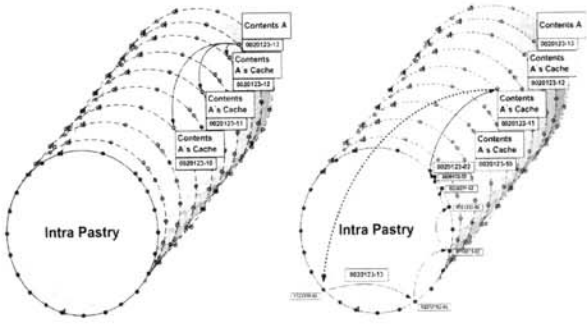



(그림 6) Doughnut의 라우팅 과정

역 내에서 키 값의 상위 128-N 비트와 가장 근접한 아이디를 갖는 노드임을 알게 되면 이 후 메시지는 Inter-Pastry 라우팅 테이블을 이용하여 라우팅이 이루어진다. Doughnut의 라우팅 알고리즘의 의사 코드(Pseudo Code)는 <표 2>에 나타나며 라우팅 과정은 (그림 6)에 보인다.

### 3.4 자기 조직화(Self-Organization)

새롭게 오버레이 네트워크에 참여하는 노드는 Intra-Overlay 초기화와 Inter-Overlay 초기화 과정을 거친다. Intra-Overlay 초기화는 새로운 노드 (X,y)(Intra-Pastry 아이디, Inter-Pastry 아이디)가 지역 내 다른 노드 (A,y)를 발견하고 자신을 (A,y)에게 알리게 된다. 그 후 (A,y)는 키 값으로 (X,y)의 노드 아이디를 사용하여 메시지를 라우팅 한




(그림 7) Doughnut의 캐시 분포 및 동작

다. 새로운 노드 (X,y)는 노드 (A,y)부터 (X,y)까지의 경로에서 만난 노드로부터 Intra-Pastry 라우팅 테이블의 n번째 행을 얻는다.

Inter-Overlay 초기화는 (A,y)부터 (X,y)까지의 경로에서 만난 노드 중 마지막에 만난 노드의 Inter-Pastry 라우팅 테이블을 이용하게 된다. 마지막에 알게 된 노드의 아이디가 (W,y)이고 새로 알고자 하는 Inter-Pastry 라우팅 노드가 (X,h)이면 노드 (X,y)는 (W,y)의 Inter-Pastry 데이터인 (W,h)에게 자신을 알리게 된다. (W,h) 노드는 노드 (X,h)를 키 값으로 하여 메시지를 라우팅 하며 (W,h)부터 (X,h)까지의 경로에서 마지막에 만난 노드의 정보는 (X,y)의 Inter-Pastry 라우팅 테이블에 기록된다.

### 3.5 캐싱

Doughnut에서 캐쉬는 Inter-Pastry 아이디만을 고려하여 이루어진다. 콘텐츠의 요구가 빈번할 경우 콘텐츠를 소유한 노드는 Inter-Pastry 아이디에서 공통의 프리픽스 길이가 같은 지역을 선택하게 되며 이때 지역 내 콘텐츠를 캐쉬하는 노드는 콘텐츠를 소유한 노드와 Intra-Pastry 아이디가 같거나 가장 근접한 노드가 된다. 콘텐츠의 요구가 더욱 증가할 경우 공통 프리픽스를 줄여가며 더욱 많은 지역에 이를 배포하게 된다. 콘텐츠 캐쉬의 방법은 (그림 7)의 왼쪽에 나타난다. Inter-Pastry 아이디만을 고려한 콘텐츠 캐쉬 방법은 모든 지역에 고르게 콘텐츠 캐쉬를 배포할 수 있으므로 몇몇 지역에 콘텐츠 캐쉬가 집중됨을 방지 할 수 있다. 또한 Intra-Pastry 아이디가 근접한 노드들로 이루어진 Inter-Pastry 라우팅 테이블은 캐쉬된 콘텐츠가 해당 지역뿐만 아니라 다른 지역의 콘텐츠 요구 또한 처리 할 수 있도록 돕는다. 이는 오버레이 네트워크상에서 발생한 콘텐츠 요청이 모든 콘텐츠 캐쉬에게 분배되도록 하기에 효율적인 캐쉬 사용이 가능토록 한다. 캐쉬된 콘텐츠가 콘텐츠 요청을 처리하는 것은 (그림 7)의 오른쪽에 나타난다.

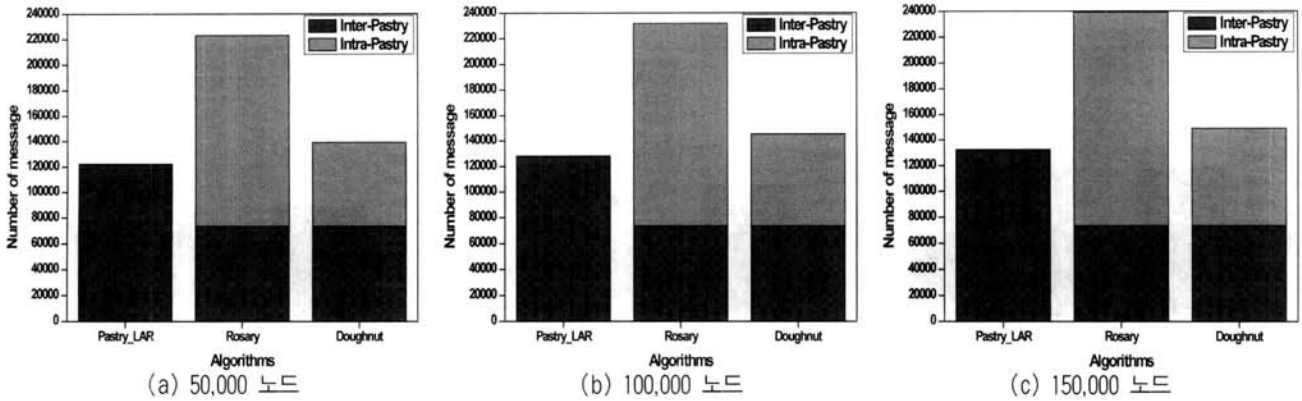
## 4 실험 및 토론

### 4.1 실험 환경

실험은 시뮬레이터(Pastry, Rosary, Doughnut, Pastry_LAR, Rosary_LAR, Doguhnut_LAR 알고리즘을 직접 구현)를 이용하여 수행 되었으며 실험에 사용된 상수와 변수는 <표 3>과 같다. 기존의 Pastry 시뮬레이터는 노드 수의 제약으로 인하여 지역적으로 나뉜 많은 노드 수를 지원하지 못하였기에 라우팅 및 콘텐츠 요청만을 세는 시뮬레이터를 제작하게 되었다. 노드 아이디는 28 bit로 제한하였고 그 중 Intra-Pastry 아이디는 20 bit, Inter-Pastry 아이디는 8 bit로 하였다. 지역은 256개이며 b=4, L=16 으로 고정되었다. 실험에 사용된 노드 수는 5만, 10만, 15만이며 콘텐츠는 100개를 실험에 사용하였다. 실험은 노드수의 증가에 따라 콘텐츠 요청 시 발생하는 총 메시지 개수, 각 피어가 처리한 메시지 개수를 측정하였다.

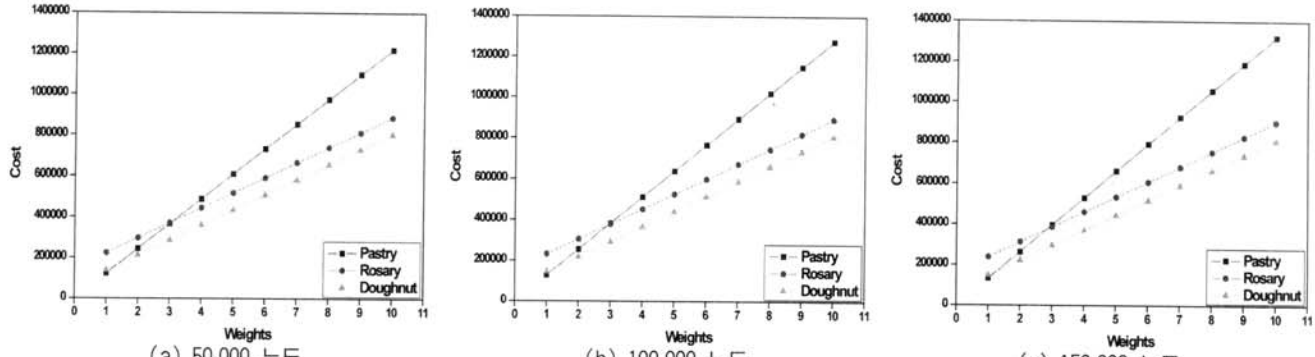
〈표 3〉 실험에 사용된 상수 및 변수

| | | |
|---|---|---|
| 상수 | 노드 아이디 | 0 ~ 228-1 |
| | 2b | 16 (b = 4) |
| | \|L\|/2 | 8 (L = 16) |
| | 파일 수 | 100개 |
| | 지역성 | 8bit (256 Intra-Pastry) |
| 변수 | 노드 수 | 50,000, 100,000, 150,000 |
| | 콘텐츠 당 쿼리 수 | 256개, 1024개 |
| | 콘텐츠 캐쉬 수 | 16개, 256개 |




(a) 50,000 노드 (b) 100,000 노드 (c) 150,000 노드

(그림 8) 각 알고리즘의 총 메시지량




(a) 50,000 노드 (b) 100,000 노드 (c) 150,000 노드

(그림 9) 가중치가 적용된 총 메시지량

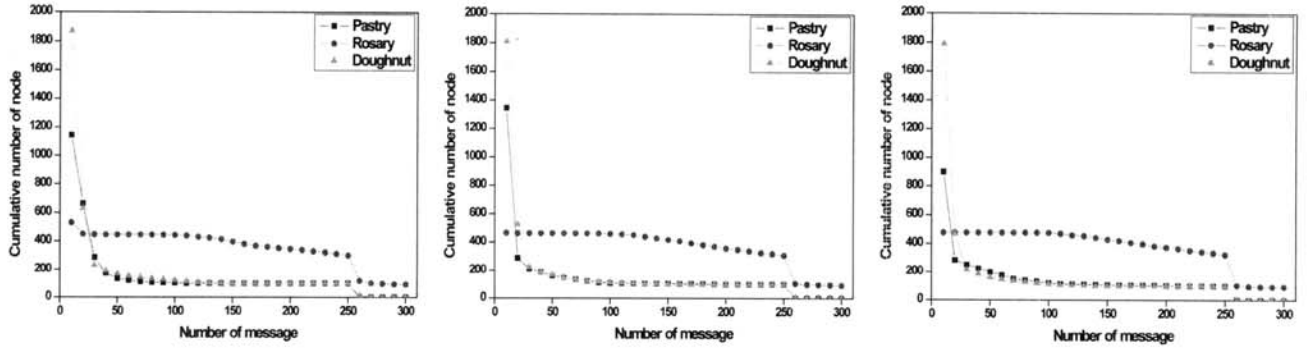
### 4.2 실험 결과

(1) 지역성

지역성의 실험은 50,000, 100,000, 150,000개의 노드로 구성된 오버레이 네트워크에서 100개의 콘텐츠를 배포하고 각 콘텐츠마다 256개의 요청을 임의로 선택된 노드에서 전송하였다.

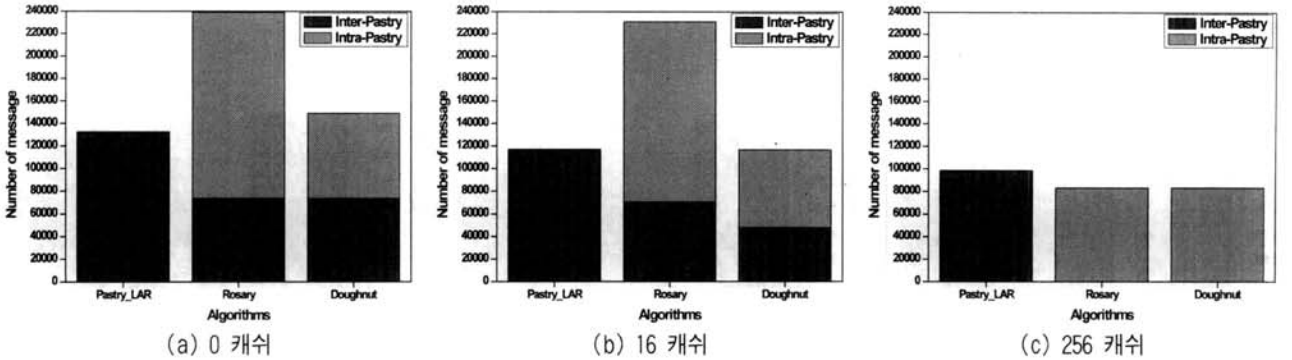
*A. 메시지 수*

(그림 8)은 각 알고리즘이 구성한 오버레이 네트워크에서 발생한 총 메시지의 개수를 보여준다. 그림에서 Pastry의 메시지의 대부분은 지역 사이를 이동하는 Inter-Pastry 메시지임을 나타내며 지역성이 고려되지 않은 Pastry 알고리즘이 비효율적임을 보여준다. Rosary와 Doughnut은 같은 양의 Inter-Pastry 메시지를 발생 시키지만 Intra-Pastry 메시지 양에서 큰 차이가 있음을 볼 수 있다. 이는 Rosary의 경우 3단계의 라우팅(Intra, Inter, Intra)과 지역에서 콘텐츠가 발견되지 않았을 때 루트 노드로 전송하는 메시지가 필요하지만 Doughnut의 경우 2단계의 라우팅(Intra, Inter)만을 필요로 하기 때문이다. Pastry와 Doughnut의 총 메시지를 비교할 때 Doughnut은 Pastry보다 총 메시지가 약간 증가 한다. 이는 Inter-Pastry 라우팅 시 콘텐츠 키의 128-N bit와 현재 노드의 Intra-Pastry 아이디가 달라질 수 있기 때문이다. 이



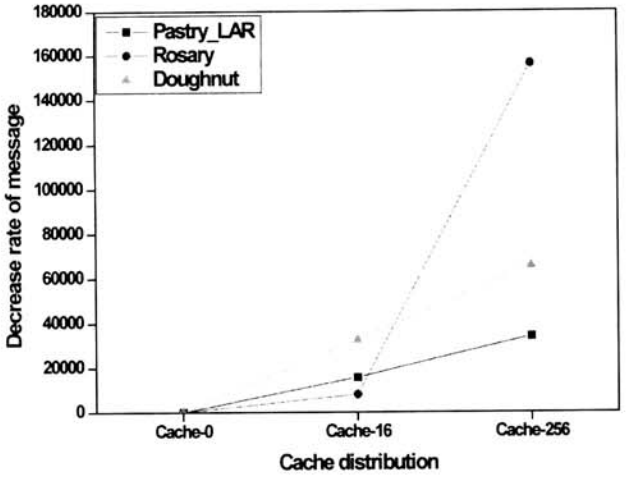

(a) 50,000 노드 (b) 100,000 노드 (c) 150,000 노드

(그림 10) 부하에 따른 노드의 누적 수




(a) 0 캐쉬 (b) 16 캐쉬 (c) 256 캐쉬

(그림 11) 캐쉬 배포에 따른 총 메시지 양




(그림 12) 각 알고리즘의 메시지 감소량

때 Doughnut은 키의 128-N bit와 가장 근접한 Intra-Pastry 아이디를 갖는 노드로 메시지를 전송하게 되므로 추가적인 Intra-Pastry 메시지가 필요하게 된다. 피어의 수가 증가 할수록 Rosary의 메시지 증가량은 Pastry와 Doughnut보다 높으며 Pastry와 Doughnut은 비슷한 증가량을 보인다.

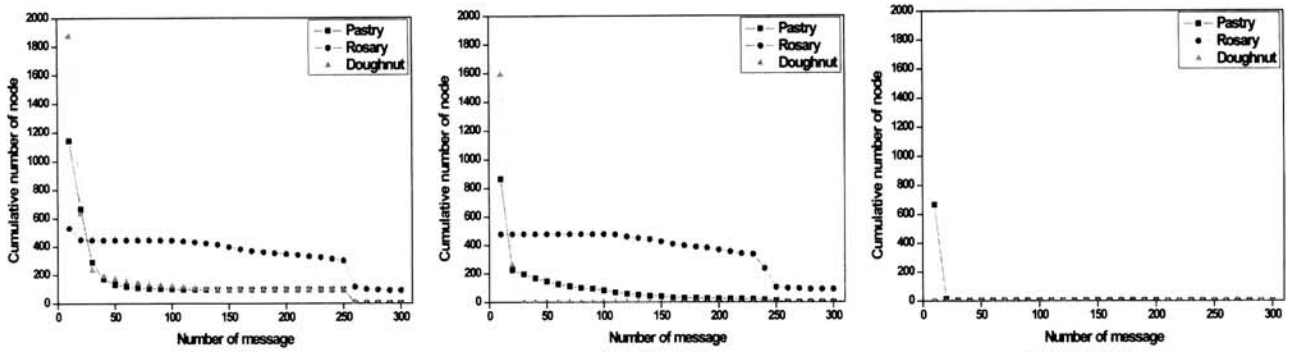
(그림 9)는 (그림 8)의 결과로부터 Inter-Pastry 메시지에 가중치를 적용했을 경우를 보여준다. Inter-Pastry 메시지와 Intra-Pastry의 비용이 같다고 할 때 Pastry 알고리즘이 가장 적은 비용을 갖으나 실제 지역 간의 메시지가 지역 내의 메시지보다 비용이 크다는 점을 고려하면 Doughnut은 Inter-Pastry 메시지가 Intra-Pastry 메시지보다 2배보다 클 경우, Pastry보다 더 효과적임을 볼 수 있다.

*B. 노드 부하*

(그림 10)은 파일 100개에 각각 256개씩의 쿼리를 보냈을 경우 메시지를 처리한 노드들의 누적치를 보여준다. 유사한 그래프를 보이는 Pastry와 Doughnut의 경우 대부분의 노드들이 20개미만의 메시지를 처리했으며 30개 이상의 메시지를 처리한 노드는 실제 콘텐츠 키값과 근접한 노드 아이디를 갖기에 콘텐츠 요청을 수신하게 되는 노드들뿐이다. 그러나 Rosary의 경우 몇몇의 특정 노드들에게 부하가 집중됨을 볼 수 있다. 이는 콘텐츠를 소유한 노드 100개와 각 지역의 루트노드(256개)가 부하가 많이 걸림을 설명하며 Rosary의 부하 불공평성을 보여준다.

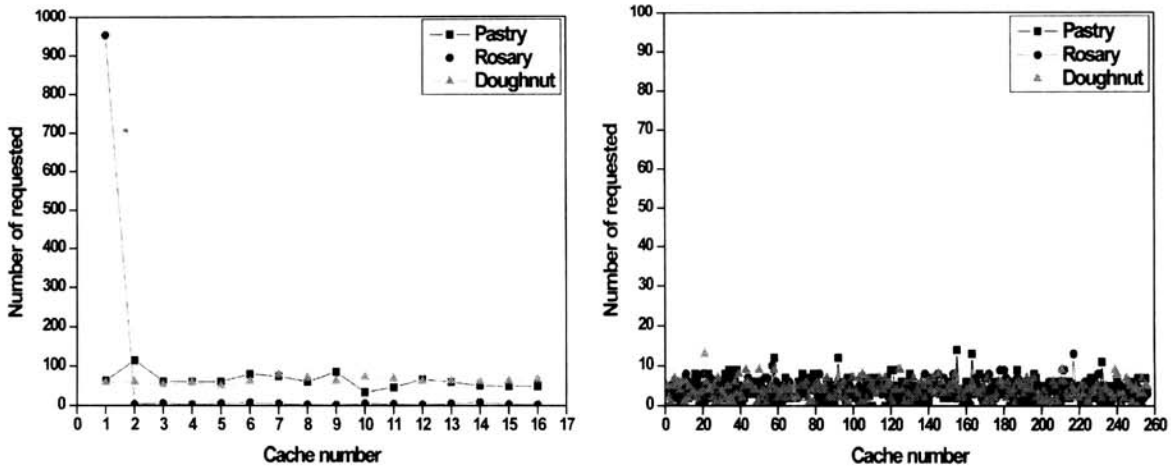
(2) 캐싱

콘텐츠 캐쉬의 실험은 150,000개의 노드로 구성된 오버레이 네트워크에 100개의 콘텐츠를 배포하고 각 콘텐츠마다 1024개의 요청을 임의로 선택된 노드에서 요구함으로써 발

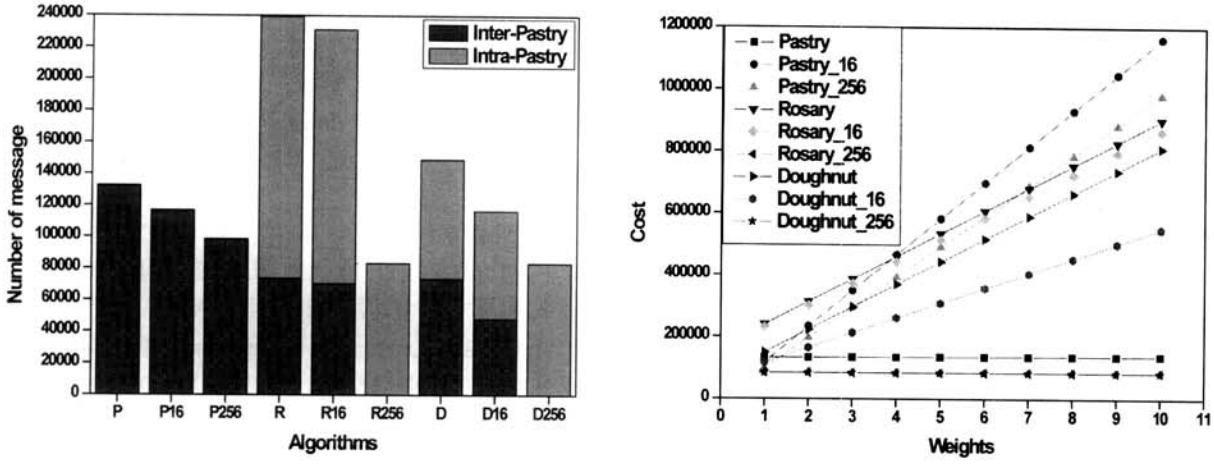



(a) 0 캐쉬 (b) 16 캐쉬 (c) 256 캐쉬

(그림 13) 캐쉬 배포에 따른 노드의 누적 수




(그림 14) 콘텐츠 캐쉬를 소유한 노드가 받은 요청 수




(그림 15) 총 메시지 양 및 가중치가 적용된 메시지 양

생하는 메시지 양을 측정 하였다.

*A. 메시지 수*

(그림 11)은 콘텐츠 캐쉬 범위를 넓혀가며 메시지의 수를 측정한 것이며 콘텐츠 캐쉬 범위가 넓어질수록 오버레이 네트워크상의 메시지 량이 줄어듦을 확인할 수 있다. Pastry_LAR의 경우 지역성을 고려한 콘텐츠 캐쉬의 배포가 아니기에 캐쉬의 범위가 넓어지더라도 Inter-Pastry 메시지가 대부분임을 보인다. 모든 지역에 캐쉬가 배포될 경우 Doughnut과 Rosary는 동일한 수의 콘텐츠 캐쉬를 배포한 Pastry_LAR 보다 오버레이 네트워크상의 총 메시지가 적음을 볼 수 있다. (그림 12)는 콘텐츠 캐쉬 배포에 따른 메시지 감소량을 보여준다. Pastry_LAR과 Doughnut은 2b에 비례하여 감소하며 Doughnut은 Pastry_LAR 보다 더 높은 메시지 감소율을 보여준다. Rosary의 메시지 감소율은 콘텐츠 캐쉬가 배포된 지역 수에 비례하여 감소하기에 적은 지역에 콘텐츠 배포 시 Pastry_LAR, Doughnut보다 효과적이지 않다.

*B. 노드 부하*

(그림 13)은 (그림 11)의 실험에서 각 노드들의 부하를 측정한 것이다. 콘텐츠 캐쉬 범위가 넓어질수록 노드들의 부하가 줄어듦을 확인 할 수 있다. 모든 지역에 콘텐츠가 배포 되었을 경우 Rosary의 루트 노드는 다른 지역으로 메시지를 전송하는 역할을 피할 수 있기에 루트 노드의 부하는 줄어들게 된다.

*C. 노드 부하 (컨텐츠 캐쉬)*

(그림 14)는 콘텐츠 캐쉬를 가진 노드들이 콘텐츠의 요청을 받은 수를 보여준다. Rosary의 경우 지역 내 콘텐츠는 해당 지역의 노드들만을 위해 서비스를 제공하기에 콘텐츠를 소유한 기존 노드의 부하는 공평하게 분담되지 않는다. Pastry_LAR의 경우 16개의 콘텐츠 캐쉬 배포 시 Leaf Set의 영향으로 콘텐츠를 소유한 노드의 부하는 캐쉬를 갖은 노드보다 부하가 약간 높다. Doughnut의 경우 지역 내 콘텐츠는 다른 지역의 노드들을 위해 서비스를 제공하며 Leaf Set의 영향이 Intra-Pastry로 제한되기에 Leaf Set의 영향을 받지 않는다. 이에 Pastry_LAR보다 공평한 부하 분담을 갖게 된다. 콘텐츠 캐쉬가 모든 지역에 분배될 경우 부하는 모든 노드에 균등히 분배되며 Rosary와 Doughnut은 Pastry_LAR보다 더 작은 표준 편차를 갖는다. 각 콘텐츠 캐쉬가 받은 콘텐츠 요청에 관한 표준 편자는 <표 4>에 나타난다.

〈표 4〉 각 알고리즘의 파일요청에 대한 표준편차

| | 16 Cache | 256 Cache |
|---|---|---|
| Pastry_LAR | 18.85 | 2.32 |
| Rosary | 237.34 | 2.03 |
| Doughnut | 6.97 | 2.10 |

(3) 지역성 및 캐슁

(그림 15)는 수행된 실험의 전체적인 비교를 보여준다. Pastry_LAR의 경우 콘텐츠 캐쉬는 Intra-Pastry 메시지의 발생에 영향을 미치지 않으며 Inter-Pastry 메시지만 줄어듦을 볼 수 있다. Rosary의 경우 콘텐츠 캐쉬가 배포될 때 이에 따른 메시지 량의 변화가 적음을 알 수 있다. Rosary의 콘텐츠 캐쉬에 따른 메시지 감소율은 콘텐츠 캐쉬가 배포된 지역의 수에 비례하여 감소하지만 Doughnut의 메시지 감소율은 라우팅 테이블의 프리픽스에 따른다. 이는 전체 지역이 아닌 적은 지역에 콘텐츠 캐쉬의 배포 시 Doughnut이 Rosary보다 메시지 감소율에서 더 효과적임을 보여준다. 모든 지역에 콘텐츠가 분배될 경우 Rosary와 Doughnut은 비슷한 성능을 가짐을 알 수 있다. 각 알고리즘의 결과에 가중치를 적용한 것은 (그림 15)의 왼쪽에 보인다.

### 4.3 토론

제안된 Doughnut 알고리즘의 장점은 크게 3가지로 요약될 수 있다. 하나는 오버레이 네트워크상에서 발생하는 메시지의 비율이 기존의 지역성을 제공하는 알고리즘보다 확

연히 줄어든다는 점이며 두 번째로 Doughnut의 노드의 부하는 모든 노드에게 공평히 분배된다는 점이다. 그리고 세 번째로 Doughnut의 콘텐츠 캐쉬는 해당 지역뿐 아니라 다른 지역에게도 서비스가 가능함으로써 콘텐츠 캐쉬가 효율적이며, 캐쉬를 소유한 모든 노드에게 콘텐츠 요구가 고르게 분배된다는 점이다.

제안된 방법의 단점은 지역성을 제공하는 Doughnut은 기존의 Pastry 알고리즘 보다 메시지가 약간 증가한다는 점, 그리고 초기 라우팅 테이블 작성 시 기존 Pastry 알고리즘보다 Inter-Pastry 라우팅 정보의 수집을 위해 메시지가 증가한다는 점이다. 그러나 지역 간의 메시지가 지역 내의 메시지보다 비용이 크다는 점을 고려할 때 제안된 알고리즘이 효과적임을 예상 할 수 있으며, 새로운 노드의 초기화시 Inter-Pastry 라우팅 테이블 작성을 위한 트래픽의 생성은 오버레이 네트워크에서 발생하는 메시지의 사소한 부분임을 고려할 때, 크게 성능에 영향을 미치지 않을 것으로 예상된다.

## 5. 결 론

본 논문에서는 기존의 지역성 알고리즘의 단점을 보완하는 Doughnut 형태의 구조를 가지는 오버레이 네트워크에 대해 제안하였다. 기존의 알고리즘이 고장 방지 능력 측면이 약하고, 오버레이 네트워크상에 메시지를 많이 발생 시키며, 노드 간 부하 분산이 균등하지 못한 점, 그리고 비효율 적인 캐쉬 방법과, 불균등한 캐쉬 콘텐츠의 요구 등을 사용한다는 점을 고려할 때 제안된 Doughnut 알고리즘은 이러한 문제점을 해결할 수 있음을 보였다.

향후 연구 방향으로는 지역 내 효율적인 캐쉬 분포를 적용하는 부분이다. 지역마다 인기 있는 콘텐츠가 다르다는 점을 고려 할 때 어떤 지역에서 인기 있는 콘텐츠는 다른 지역에서 인기가 없을 수 있다. 이때 지역마다 균등한 캐쉬를 생성하는 단순 지역 배포 방식은 비효율 적임을 알 수 있다. 각 지역이 인기 있는 콘텐츠의 캐쉬가 해당 지역에 효율적으로 분배 될 때 캐쉬 콘텐츠의 효율성 측면에서 높은 효과를 보일 것이다.

## 참 고 문 헌


[1] 박호진, 박광로, "P2P 기술 동향 및 홈 네트워크 응용(P2P Technology Trend and Application to Home Network)", 전자통신동향분석, 제21권, 제5호, 2006.10.

[2] 김병오, 이일우, 박호진, "분산 해시 테이블 기반 P2P 기술 동향(Trend of Distributed Hash Tables-Based P2P)", 전자통신동향분석, 제21권, 제6호, 2006.12.

[3] Napster, http://www.Napster.com

[4] Seti@home, http://www.setiathome.berkeley.edu

[5] Skype, http://www.skype.com

[6] Zoost, http://www.zoost.tv

[7] Gnutella, http://www.gnutella.com

[8] JXTA, http://www.sun.com/jxta

[9] I. Stoica, R. Morris, D. Liben-Nowell, D. Karger, M. Kaashoek, F. Dabek, and H. Balakrishnan, "Chord: A Scalable Peertopeer Lookup Service for Internet Applications", IEEE/ACM Transactions on Networking, Vol.11, pp.17-32, 2003.

[10] A. Rowstron and P. Druschel, "Pastry: Scalable, distributed object location, and routing for large-scale, persistent peer-to-peer systems", In Proc ACM SOSP, 2001.

[11] S. Ratnasamy, P. Francis, M. Handley, and R. Karp, "A Scable Content Addressable Network", In Proc. of SIGCOMM, 2001.

[12] J. Namgung, S. Shin, S. Park, L. Lee, and D. Jeong, "Self-organizing P2P overlay network applying dynamic landmark mechanism for contents delivery network", Third ACIS International Conference on IEEE, pp.317-324, 2005.

[13] David R. Karger, Matthias Ruhl, "Simple Efficient Load-Balancing Algorithms for Peer-to-Peer Systems ", Theory of Computing Systems ACM SPAA, Vol.39, No.6, pp.787-804, 2006.

[14] Z. Li, G. Xie, and Z. Li, "Locality-Aware Consistency Maintenance for Heterogeneous P2P Systems", IEEE Parallel and Distributed Processing Symposium, pp.1-10, 2007.

[15] V. Gopalakrishnan, B. Silaghi, B. Bhattacharjee, and P. Keleher, "Adaptive replication in peer-to-peer systems", IEEE Distributed Computing Systems, pp.360-369, 2004.

[16] H. Shen and C. Xu, "Locality-Aware and Churn-Resilient Load-Balancing Algorithms in Structured Peer-to-Peer Networks", IEEE Transactions on Parallel and Distributed Systems, Vol.18, pp.849-862, 2007.



**김 명 원**

e-mail : king@q.ssu.ac.kr

2006년 숭실대학교 정보통신전자공학부 (학사)

2006년 3월~현 재 숭실대학교 정보통신전자공학부 석사과정

관심분야 : 임베디드 컴퓨팅

## 곽 후 근

e-mail : gobarian@q.ssu.ac.kr
1996년 호서대학교 전자공학과(학사)
1998년 숭실대학교 전자공학과(석사)
1998년~2006 숭실대학교 전자공학과(박사)
1998년 8월~2000년 7월 (주) 3R 부설 연구소 주임 연구원
2006년 3월~현 재 숭실대학교 정보통신전자공학과 postdoc
관심분야 : 네트워크 컴퓨팅 및 보안

## 정 규 식

e-mail : kchung@q.ssu.ac.kr
1979년 서울대학교 전자공학과(공학사)
1981년 한국과학기술원 전산학과(이학석사)
1986년 미국 University of Southern California(컴퓨터공학석사)
1990년 미국 University of Southern California(컴퓨터공학박사)
1998년 2월~1999년 2월 미국 IBM Almaden 연구소 방문 연구원
1990년 9월~현 재 숭실대학교 정보통신전자공학부 교수
관심분야 : 네트워크 컴퓨팅 및 보안