Rosary : CDN 시스템을 위한 구조화된 토폴러지-인식 P2P 오버레이 네트워크

정회원 신 수 영*, 남 궁 정 일**, 박 수 현*

Rosary : Topology-Aware Structured P2P Overlay Network for CDN System

Soo-Young Shin*, Jung-Il Namgoong**, Soo-Hyun Park* ***Regular Members***

요 약

최근 CAN, Chord, Pastry와 Tapestry 등과 같이 안전한 분산 시스템기반의 P2P (Peer to Peer) 오버레이 네트워크가 일반화되고 있다. 이러한 시스템은 효율적이고 결함에 허용적인 라우팅과 개체의 위치추정 및 스스로 초기화 시킬 수 있는 오버레이 네트워크가 갖는 로드 밸런싱을 제공한다. CDN (Contents Delivery Network)은 네트워크 하부구조의 중간 단계로서 궁극적으로 CP(Contents Provider)가 고객에게 전달하고자하는 멀티미디어 데이터를 포함한 컨텐츠의 효율적인 전송을 돕는다. 본 논문에서는 CDN에 적합한 토폴로지-인식 P2P 오버레이 네트워크를 새롭게 제안하였다. 제안된 기법인 Rosary에서 CDN 서버는 구조화된 오버레이 네트워크상에서 두 단계의 Inter-pastry와 Intra-pastry 라우팅을 수행한다. Rosary는 기존의 Pastry 기법을 CDN 환경에 맞도록 적응적으로 수정, 보완 및 추가하였으며 언급한 Intra-pastry 방식과 동적인 랜드마크 토폴로지 기법에 기반을 둔 세미-해싱방법을 제공하고 토폴로지-인식 오버레이 네트워크의 구성과 운용에 이용되었다. NS-2를 이용한 시뮬레이션에서 Rosary 기법은 네트워크 구성 후 안정화 된 컨텐츠 전송에서 보다 전송실패가 적고 효율적이며 견고한 것으로 증명되었다

Key Words : CDN(Contents Delivery Network), P2P network, Pastry, Overlay network

ABSTRACT

Recently, Peer-to-Peer (P2P) overlay networks like CAN, Chord, Pastry and Tapestry offer a novel platform for scalable and decentralized distributed applications. These systems provide efficient and fault-tolerant routing, object location, and load balancing within a self-organizing overlay network. Content Delivery Network (CDN) is an intermediate layer of infrastructure that helps to efficiently deliver the multimedia content from content providers to clients. In this paper, We propose a topology-aware P2P overlay network for CDN, Rosary, in which CDN servers perform Intra-Pastry and Inter-Pastry routing based on a two-level structured overlay network. This proposed system extends Pastry by adapting itself to CDN environments, where a semi-hashing based scheme for Intra-Pastry routing is introduced, and dynamic landmark technology is used to construct the topology-aware overlay network. Through simulations on NS-2, it is shown that Rosary is scalable, efficient, and flexible.

※본 연구는 정보통신부 및 정보통신연구진흥원의 대학 IT연구센터 지원사업의 연구결과로 수행되었음.(IITA-2005-C1090-0501-0010)
* 국민대학교 비즈니스 IT학부 비즈니스 정보통신 연구실 ({sy-shin, shpark21}@kookmin.ac.kr), ** greenji@naver.com
논문번호 : KICS2005-08-313, 접수일자 : 2005년 8월 13일

Ⅰ. 서 론

최근 유비쿼터스 통신의 다양한 기술이 혁신적으로 발전하면서 노드간, 서비스간, 이종망간의 연동과 통합이 가속화되고 있다[1][2]. 이미 사용자의 요구에 따라 끊김 없는 음성(Voice over IP : VoIP)이나 동영상(Video over IP), 짧은 시간에 대량의 컨텐츠를 신뢰성있게 전송할 수 있는 기술적 지원은 유비쿼터스 통신이 지향하는 목적에 가장 근접할 수 있게 하는 필수 조건이 되었다. P2P(Peer-to-Peer)를 이용한 효율적인 컨텐츠의 배포 기술인 CDN은 인터넷에서 유용하게 사용되어지고 있으며 나날이 그 중요성이 커지고 있다. 이는 앞으로 진정한 유비쿼터스 통신의 실현에서 다양한 발전으로의 교각 역할을 해 낼 것으로 보여진다. [2]

그누텔라(Gnutella)[3], 프리넷(Freenet)[4], 냅스터(Napster)[5], 모피어스(Morpheus)[6], 카자(Kazaa)[7] 등과 같은 제 1 세대 P2P 파일 공유 시스템과 최근의 제 2 세대 P2P 시스템인 CAN[8], Chord[9], Pastry[10], Tapestry[11]와 같은 분산 P2P 시스템과 그 시스템을 활용한 애플리케이션들이 활발히 연구되고 있다. 이러한 시스템들은 대규모 P2P 애플리케이션을 위한 셀프-오거나이징(Self-Organizing) 기술을 제공한다. 확장성 있고 장애에 강한 분산 해쉬 테이블을 이용하는 공통점을 갖고 있는 반면 하부 인터넷(Underlying Internet)에서 프락시머티(Proximity)를 고려하고 이용하는 방법에는 차이점이 있다[12].

Pastry와 Tapestry 기법은 각 노드들의 라우팅 테이블 내 엔트리에 해당하는 인접노드들을 선택하기 위해서 인터넷 토폴로지-인식 오버레이(Internet Topology-Aware Overlay)를 형성한다[10][11]. 본 논문에서는 Pastry를 수정하고 추가하여 CDN(Content Delivery Network) 환경에 적합하게 확장한 Rosary 오버레이 네트워크를 제안한다. 기존의 Pastry의 경우 해쉬 기반 기법을 적용하였기 때문에 사용자들의 요청이 빈번한 컨텐츠의 경우 한 노드로 네트워크의 전송 부하가 집중되는 현상이 발생할 수 있어 CDN 시스템에 적합하지 않았다. Rosary는 이러한 점을 극복하기 위하여 기존의 Pastry를 Inter-Pastry와 Intra-Pastry로 구분하여 그 토폴로지를 좀 더 CDN 환경에 맞게 수정하고 확장하였다. 즉 애플리케이션 수준의 멀티캐스팅(Application-Level Multicasting)을 위해 그룹 개념인 Intra-Pastry를 추가하여 완전 해쉬 기법(Fully Hash Scheme)이 아닌 세미 해쉬 기법(Semi Hash Scheme)을 적용한 것이다. 또한 프락시머티 이웃 선택(Proximity Neighbor Selection) 기술을 적용한 Pastry에 개선된 지리적 레이아웃(Geographic Layout) 기술인 동적 랜드마크(Dynamic Landmark) 기능을 추가하여 완전한 셀프-오거나이징 특성을 유지할 수 있게 하였다.

본 논문의 구성은 다음과 같다. 2장에서는 연구의 기반이 되는 CDN과 Pastry 기법에 대해서 설명하고 3장에서는 Pastry를 CDN 환경에 맞게 개선한 Rosary를 제안하였다. 4장에서는 Rosary에 대한 시뮬레이션 모델과 결과를 기술하였고, 마지막으로 5장 결론을 맺는다.

Ⅱ. CDN과 Pastry

CDN은 기간망과 가입자망간의 연결을 물리적인 망의 증설을 통해 계산하지 않고 병목(bottleneck)현상의 대상인 데이터 트래픽 즉, 컨텐츠를 인터넷 네트워크의 주요 지점으로 분산시킨다. 복잡한 인터넷의 미들마일(Middle-Mile)을 지나는 트래픽의 양을 줄이고 분산된 여러 서버로 컨텐츠를 분배하여 CP의 서버 부하를 줄이고 사용자에게는 최단 경로를 통한 전송속도의 향상과 컨텐츠 전송 품질의 보장을 제공할 수 있다. 최근 P2P 파일 공유 시스템의 확장성을 위하여 분산 해쉬 테이블(Distributed Hash Tables: DHTs) 기능을 지원하는 Tapestry, Pastry, Chord, CAN 등의 새로운 P2P 시스템들이 제안되었고 대규모 분산 시스템을 위한 기법으로 그 성능이 입증되었다. DHT 노드들은 오버레이 네트워크를 형성하며 각각의 노드들은 이웃노드(Neighbor- Node)의 목록과 Lookup인 키 정보를 유지하며 그 정보는 오버레이 네트워크를 통해 그 목적지로의 라우팅이 가능하게 한다. Pastry는 대표적인 오버레이 네트워크이다. Pastry는 확장성이 있고 장애에 강하며 셀프-오거나이징 P2P 기반 기법이다. 각각의 Pastry 노드는 원형의 128-bit 식별자 공간(Circular 128-bit Identifier Space)에서 유일하고 균등하며 랜덤하게 할당된 노드 ID(nodeID)를 가진다. 128-bit 키로 Pastry는 관련 메시지를 수치적으로 그 키와 가장 가까운 노드ID를 가진 라이브 노드(Live Node)로 라우팅한다. 더욱이 각각의 Pastry 노드는 이름공간(Name-Space)에서 이웃하는 노드들의 정보를 계속 유지하고 있으며 애플리케이션에 그 정보의 변화를 통지한다.

2.1 Pastry 노드와 메시지 라우팅

라우팅을 위해 노드 ID와 키는 기본적으로 2b(b

는 일반적으로 4를 갖는 설정 파라미터) 만큼의 디지트(digit) 열로 할당된다. 노드의 라우팅 테이블은 128/2b 행(row)과 2b 열(column)로 구성되며 라우팅 테이블에서 행 n 의 2b 엔트리(entry)들은 노드ID가 현재 노드의 노드ID 와 첫 n digits를 공유하는 노드의 IP 주소를 포함한다. 그림 1은 노드ID 65a1x, b = 4인 Pastry 노드가 갖는 라우팅 테이블 예이다. x는 임의의 접미사(suffix)를 나타낸다[12]. 먼저 각각의 노드들은 리프셋(Leaf-Set) 목록을 유지한다. 리프셋은 현재 노드의 ID보다 l/2 더 크고 l/2 더 작은 노드ID를 가진, 수치적으로 현재 노드의 노드ID와 가장 가까운 노드ID를 가진 l 노드들의 집합이다. 일반적인 l 값은 대략 8 * log16N 이다. 리프셋 목록은 신뢰성 있는 메시지 전송을 보장하며 애플리케이션 개체의 복제시 저장을 위해 사용된다.

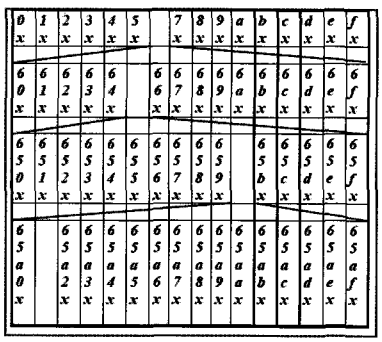


그림 1. Pastry 노드의 라우팅 테이블

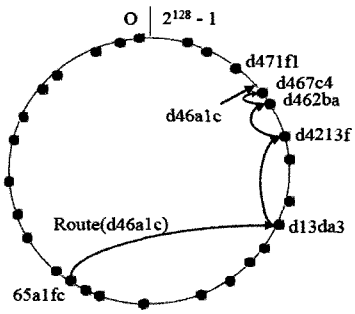


그림 2. Pastry 노드에서의 메시지 라우팅

각각의 라우팅 단계에서 노드는 현재 노드가 공유하는 프리픽스(prefix)보다 최소한 1 디지트 (또는 b 비트) 더 큰 프리픽스에 키를 공유하는 노드ID를 가진 노드로 메시지를 전송하려고 한다. 그런 노드가 라우팅 테이블에서 발견되지 않으면 그 노드와 근접한 키를 이용하여 프리픽스를 공유하는 노드ID를 가진 노드로 포워딩한다. 그림 2는 키 d46a1c를 가지고 노드 65a1fc로부터 메시지를 라우팅하는 예이다[12]. 표 1은 키 d 를 가진 메시지가 노드ID a 를 가진 노드에 도착했을 때 수행되는 Pastry 라우팅 절차이다[12]. R_l^i 은 열 i와 행 l에서 라우팅 테이블 R 의 엔트리를 의미한다. Li는 리프셋 L에서 i 번째 가장 가까운 노드ID이며 네거티브(negative)/파지티브(positive) 인덱스는 ID 공간 내 로컬 노드에서 각각 반시계방향/시계방향을 나타낸다. L-l/2 와 Ll/2는 로컬 리프셋의 경계(edge)에 있는 노드들이다. dl은 키 d에서 l's 디지트를 나타낸다. shl(a, b)는 디지트로 a와 b 중에 공유되는 프리픽스의 길이이다.

표 1. Pastry 라우팅 절차

```
if (d.isBetween(L-l/2, Ll/2))
   // d is within range of local leaf set (mod 2^128)
   forward to Li, s.t. |d - Li| is minimal;
else
   // use the routing table
   Let l = shl(d, a);

   if (R_l^dl exitsts and is live)
      forward to R_l^dl ;
   else
      // rare case
      forward to t ∈ L ∪ R, s.t.
         shl(t, d) ≥ l,
         |t - d| < |a - d|
```

2.2 Pastry 로컬리티와 오버레이 유지

Pastry는 프락시머티 이웃 선택 기술을 사용한다. 가장 근접한 인접노드를 오차 없이 정확하게 참조하도록 라우팅 테이블 엔트리를 선택하는 것은 대규모 시스템에서는 비용이 많이 든다. 그렇기 때문에 Pastry는 라우팅 테이블 엔트리들이 반드시 가장 가까운 것은 아니고 어느 정도 가깝다는 것을 보장할 수 있는 휴리스틱(heuristic) 값을 사용한다. Pastry 기법에서는 각각 노드의 라우팅 테이블들이 불변의 특성들을 갖게 되므로 임의의 노드 X 의 라우팅 테이블에서 각각의 엔트리는 프락시머티 메트릭에 따라 적절한 노드ID를 가진 모든 라이브 Pastry 노드들 중에서 X 와 근접한 노드를 참조하게 되고 네트워크 로컬리티에 관하여 Pastry의 명확한 특징을 유도할 수 있다. 새로운 노드가 Pastry 오버레이 네트워크에 조인(join)할 때 노드ID X 를 가진 새 노드

는 기존의 Pastry 노드 A 에 반드시 접촉해야 한다. 그러면 A 는 키 값으로 그 ID X 를 사용하여 메시지를 라우팅한다. 그리고 새 노드는 A 부터 X 까지의 경로를 따라 만난 노드로 부터 그것의 라우팅 테이블의 n번째 행을 얻는다. 노드 접속을 실패할 경우 메시지를 수치적으로 더 가까운 노드ID를 가진 또 다른 노드로 라우팅한다. 만일 다운스트림(downstream) 노드가 메시지 키의 다음 디지트와 일치하는 라우팅 테이블 엔트리를 가지고 있다면 그것은 자동적으로 그 엔트리의 업스트림(upstream) 노드에게 알린다.

라우팅 테이블 유지(Routing table maintenance)와 전체 경로 품질의 저하를 막기 위해 각각의 노드는 주기적으로 라우팅 테이블 유지 작업을 수행한다. 로컬 노드의 라우팅 테이블 각각의 행에 대해 다음의 절차를 수행하게 되며 먼저 각각의 행에서 랜덤하게 엔트리를 선택하고 관련 노드에서 그 노드의 상응하는 라우팅 테이블 행의 복사를 요청한다. 그리고 그 행 내 각각의 엔트리가 로컬 라우팅 테이블 내 상응하는 엔트리와 비교한다. 만일 비교값이 다르면 그 노드는 양쪽 엔트리들에 대한 거리를 검사하고 그 자신의 라우팅 테이블에 가장 가까운 엔트리를 설정한다.

새롭게 네트워크에 조인하는 Pastry 노드는 유지하고 있는 인접노드의 위치 정보가 없다. 그래서 인접노드로부터 정보를 얻는 것을 대신하여 일단 아웃오브밴드(out-of-band) 방식을 통해 임의의 Pastry 노드의 정보를 얻는다. 그리고 인접노드를 발견하는 알고리즘은 시드(seed)되는 리프셋(leaf set) 내 노드들의 위치가 네트워크 상에 균일하게 분산되어져야 한다는 특성을 이용해서 가장 가까운 리프셋 멤버를 발견했을 때 라우팅 테이블의 거리 특성을 이용하여 조인하는 노드의 위치에 더 가깝게 급격히 이동하게 된다. 각각의 레벨에서 가장 가까운 노드를 선택하고 그것에서 다음 레벨에 이르는 과정이 계속 반복되어 가장 가까운 값을 얻게 되는 것이다. 표 2는 인접노드 로케이팅(nearby node locating)을 위한 알고리즘을 의사코드화 한 것이다[12]. 여기서 시드는 조인하는 노드에게 잘 알려진 Pastry 노드이다.

2.3 Pastry 기법의 문제점

기존의 Pastry는 완전한 해쉬 기법을 적용하였기 때문에 컨텐츠를 특정 Pastry 노드 한 곳에만 배포를 하게 된다. 이 방법은 사용자들의 요청이 빈번한 특정 자료가 존재하는 경우에 부하가 한 노드로 집중되는 현상이 발생 시킬 수 있어 CDN 시스템에 적합하지 않다. 또한 Pastry는 키에 대한 노드ID 프리픽스(prefix)를 기반으로 라우팅을 하기 때문에 키 값과 다른 노드ID를 가지면서도 현재 노드와 인접해 있는 노드들이 있을 경우 그 전송 대상에서 제외될 수 있다. 이는 최악의 경우 실제로 인접한 노드의 절반 이상이 최적의 전송 경로를 갖지 못하는 상황을 발생시킬 수 있다. 완전한 해쉬 기법은 분산처리에 취약하다는 단점을 가지며 중앙집중식의 지리적 레이아웃 랜드마크 서버(Landmark Server)는 컨텐츠를 적절히 나누어 논리적인 네트워크를 구성 관리해야 하는 CDN 시스템으로서의 한계가 분명히 존재한다. 본 논문에서는 Pastry의 단점을 해결하기 위해 Rosary 기법을 제안하게 되었다.

표 2. 인접노드 검색 알고리즘

```
discover(seed)
  nodes = getLeafSet(seed)
  forall node in nodes
    nearNode = closerToMe(node, nearNode)
  depth = getMaxRoutingTableLevel(nearNode)
  while (depth > 0)
    nodes = getRoutingTable(nearNode, depth--)
    forall node in nodes
      nearNode = closerToMe(node, nearNode)
  end while
  do
    nodes = getRoutingTable(nearNode, 0)
    currentClosest = nearNode
    forall node in nodes
      nearNode = closerToMe(node, nearNode)
  while (currentClosest != nearNode)
  return nearNode
```

Ⅲ. Rosary

Rosary는 기존의 Pastry 기법을 기반으로 하여 CDN 시스템에 적합하게 확장한 구조화된 오버레이 네트워크이다. Rosary는 2.3절에서 언급한 Pastry의 단점을 극복하기 위하여 기존의 Pastry를 Inter-Pastry와 Intra-Pastry로 구분하여 그 토폴로지를 좀 더 CDN 환경에 맞게 확장하였다. 즉 애플리케이션 수준 멀티캐스팅이 가능하도록 그룹인 Intra-Pastry를 추가하여 완전한 해쉬 기법이 아닌 세미 해쉬 기법을 적용하였으며 Pastry가 적용한 프락시머티 이웃 선택 기술에 지리적 레이아웃(Geographic Layout) 기술을 적용시켰다. 지리적 레이아웃의 랜드마크 서버(Landmark Server)와는 달리 Rosary는 노드 조인 시 RTT를 측정하여 인접한 노드를 이웃셋(Neighborhood-Set)으로 두고 랜드마크 서버의 역할을 하게한다. 그래서 완전한 셀프 오거나이징

(Self-Organizing)의 특성을 유지하면서도 Rosary 공간의 Inter-Pastry 엔트리들이 중요한 컨텐츠를 적절히 분산하여 갖도록 하고 핫스팟을 유발하는 불균형을 개선하였다.

3.1 Inter-Pastry와 Intra-Pastry

Inter-Pastry는 루트노드들로 구성되어지며 기존의 Pastry의 특징을 유지한다. 루트노드는 CDN 시스템에 있어서 각 지역을 담당하는 서버(regional edge server)의 역할을 수행하는 노드라 할 수 있으며 Intra-Pastry의 관리 및 유지를 책임지는 프락시 서버의 역할도 수행한다. 루트노드는 컨텐츠가 존재하는 원본 서버(origin server)와의 통신을 통해 자신이 관리하는 Intra-Pastry에 가입된 리프노드(leaf node)로 컨텐츠 배포를 담당하며 해당 리프노드가 컨텐츠를 효율적으로 전송 할 수 있도록 네트워크 부하를 분산시키는 역할을 수행한다. 이렇게 하여 원본 서버로의 과도한 부하의 집중을 방지할 수 있다. Intra-Pastry는 리프노드들로 구성되어지며 하나의 루트노드에 의하여 관리되어 진다. 리프노드는 각 지역의 경계 서버(Edge Server)로서 그 지역과 인접한 유저들의 컨텐츠 요청에 대해 컨텐츠를 직접 전송하는 역할을 수행하며 루트노드의 장애 등으로 루트노드가 존재하지 않게 되면 같은 Intra-Pastry에 가입된 모든 리프노드들 중에서 가장 성능이 좋은 노드가 대체 서버로서의 역할을 수행하게 된다. 그림 3은 제안된 Rosary 토폴로지를 도식화한 것이다.

컨텐츠의 배포 및 전송의 경우 원본 서버로부터 컨텐츠의 배포 요청이 루트노드에 도착하면 각각의 루트노드는 자신에게 조인되어 있는 Intra-Pastry의 모든 리프노드에게 컨텐츠를 배포한다. 또한 Intra-Pastry 내에서 자신과 인접한 루트노드에게도 컨텐츠를 배포하게 된다. 만약 사용자가 요청한 컨텐츠가 Intra-Pastry내에 존재하지 않으면 루트노드는 인접한 다른 루트노드에게 컨텐츠의 배포를 요청하며 실패 시 원본 서버에게 컨텐츠를 요청하게 된다. 노드ID 할당(NodeID Assignment)을 위해서 Rosary 노드의 유일한 식별자인 노드ID는 기존 128 bits Pastry 노드ID를 확장한다. Rosary 노드ID는 크게 Inter-Pastry 노드ID 와 Intra-Pastry 노드ID 부분으로 구성되어진다. Inter-Pastry 노드ID는 Inter-Pastry 상에서의 라우팅을 위해 사용되어지며, Intra-Pastry 노드ID는 Intra-Pastry 상에서의 라우팅을 위해 사용되어진다. 그림 4는 Rosary의 노드ID 구조를 나타낸다.

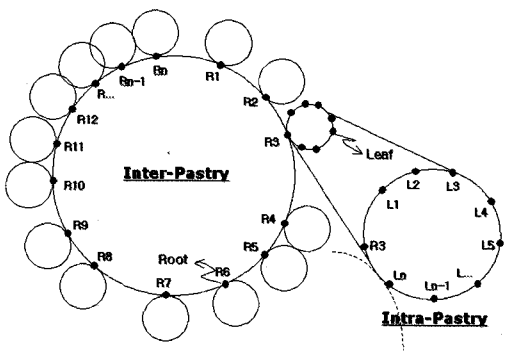


그림 3. Rosary 토폴로지

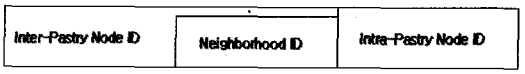


그림 4. Rosary 노드ID 구조

Inter-Pastry 노드ID의 후위 n bits는 이웃ID (neighborhood ID)로 사용되어지고 나머지는 임의로 할당되어진다. 후위 n bits 의 영역에 저장된 이웃ID는 루트노드들이 언더레이 네트워크(Underlay Network)인 인터넷상에서 물리적으로 인접한 노드임을 의미한다. 이웃ID는 RTT를 기반으로 한 거리 측정 결과 값이 미리 정해진 조건을 만족할 경우 인접노드로서 부여되는 ID이며 새로운 노드가 Rosary 네트워크에 조인 할 때 결정된다. Inter- Pastry 노드ID들은 이름공간상에서 잘 분포가 되게 임의로 할당 되어져야 하는데 그 이유는 서로 인접한 노드들이 이름공간상에서 어떤 한 부분에 집중되어 있을 경우 네트워크 장애로 인한 오버레이 네트워크의 분할이 발생할 수 있기 때문이다. 새로 조인하는 노드가 기존 Inter-Pastry의 리프노드나 소속된 Intra-Pastry의 루트노드가 될 때 인접한 Intra-Pastry가 존재하면 Inter-Pastry 노드ID 부분에서 이웃ID를 제외한 프리픽스 부분을 공유하게 된다. 이러한 이웃ID를 노드들이 유지하게 되므로 근접한 노드들끼리 이름공간 상에서 가까이 있게 되어 좀 더 효율적인 라우팅을 할 수 있게 된다. Intra-Pastry 노드ID는 기존의 Pastry의 ID 할당 체계에 따라 할당된다.

3.2 Rosary 캐쉬의 유지와 관리

동일 Intra-Pastry에 가입된 노드들은 모두 동일한 최신의 Rosary 캐쉬 목록(cache list)을 유지한다.

루트노드는 주기적으로 자신의 존재와 모든 상태 정보를 자신의 Intra-Pastry에 가입된 모든 리프노드에게 통보하며 이를 수신한 모든 리프노드들은 자신의 캐쉬 목록을 갱신한다. 또한 리프노드도 주기적으로 자신의 존재와 상태 정보를 루트노드에게 보고하게 되며 루트노드는 리프노드들로부터 받은 상태 정보(노드의 CPU 속도, OS 타입 등의 리소스 정보)를 기반으로 루트노드가 되기에 적합한 순서로 노드 목록을 갱신하며 리프셋의 정보도 갱신하게 된다. 루트를 통한 전송 실패가 발생할 경우 캐쉬 목록에서 가장 우선 순위가 높은 노드가 루트로 선정된다. 새로 루트가 된 노드는 자신이 속한 Intra-Pastry 내 모든 리프노드뿐만 아니라 이웃셋의 루트노드들에게 즉시 갱신 메시지를 전파한다.

이웃셋 갱신 알고리즘에서 각 Intra-Pastry의 루트노드들은 주기적으로 이웃셋을 갱신한다. 좀 더 효율적으로 인접 루트노드를 선정하기 위해서 이웃셋에는 프락시머티 메트릭에 따른 정보(RTT, Hop 카운트 등)를 함께 보관하게 된다. 루트노드의 변경 시 즉시 이웃셋의 모든 루트 노트들에게 갱신 메시지를 통보하며 이 메시지를 수신한 루트노드는 자신의 이웃셋을 수정하고 자신의 이웃셋에 포함된 모든 루트에게도 갱신 메시지를 포워딩한다.

리프셋 갱신 알고리즘에서 루트노드는 주기적으로 리프노드들로부터 받은 상태 정보를 기반으로 리프셋의 정보를 갱신한다. 복제 전략(Replication Strategies)에서 기존의 Pastry가 가지는 해쉬 기반 기법에서는 컨텐츠가 동일한 Intra-Pastry 내에서 하나의 리프노드에서만 존재하게 된다. 그 컨텐츠의 접속빈도가 높아질 경우 하나의 노드로 많은 요청이 집중될 수 있다. 이러한 현상을 막기 위해서 인기있는 컨텐츠 복제(Popular Content Replication) 개념을 추가했다. 인기 있는 컨텐츠 복제는 컨텐츠 요청 수(Content Request Number)를 유지하여 일정 수치를 넘을 경우에 해당 컨텐츠를 동일 Intra-Pastry 내 모든 리프노드의 저장소에 배포, 저장하여 요청 시 전송할 수 있게 한다. 표 3은 Rosary 캐쉬 목록을 갱신 알고리즘 이다.

Rosary 캐쉬 서버는 새로운 Rosary 노드가 처음 Rosary 오버레이 네트워크로 조인할 경우 진입 초기 연결 지점 문제(Initial Connection Point Problem)를 해결해주기 위해서 존재한다. 새로운 Rosary 노드는 잘 알려진 Rosary 캐쉬 서버로 루트노드 목록을 요청하게 되고 Rosary 캐쉬 서버는 Inter-Pastry 내 루트노드들의 목록을 반환한다. 이 정보를 기반으로 새로운 노드는 Rosary 오버레이 네트워크에 조인할 수 있다. Inter-Pastry의 모든 루트노드들은 자신의 존재를 주기적으로 캐쉬 서버에 통보하며 이러한 과정을 통해 캐쉬 서버는 최신의 정보를 유지할 수 있다.

표 3. Rosary 캐쉬 목록 갱신 알고리즘

```
캐쉬목록 타이머에 의해 주기적으로 수행된다.
updateCachList(msg_type)
 switch msg_type is
   ROOT_CACHE_S :
    leaf_nodes = getLeafSet()
    cache_list = getCacheList()
    root_state = getState()
    broadcastCacheList(leaf_nodes, root_state, list)
   ROOT_CACHE_R :
    rec_leaf_nodes = getRecLeafSet()
    rec_leaf_states = getRecLeafStates()
    recalculateCacheList(rec_leaf_nodes,
                                    rec_leaf_states)
    updateLeafSet(UP_RFSET_R)
   LEAF_CACHE_S :
    root_node = getRootNode()
    cache_list = getCacheList()
    leaf_state = getState()
    forwardCacheList(root_node, leaf_state)
   LEAF_CACHE_R :
    setCacheList(cache_list)
```

3.3 Rosary 노드의 조인

Rosary 노드의 상태는 Inter-Pastry 에 조인된 루트노드와 Intra-Pastry에 조인된 리프노드로 분류할 수 있다. 루트노드는 이웃셋(Neighborhood-Set)을 가지며 프락시머티 메트릭(Proximity Metric)에 의해 서로 인접한 노드인 M 개의 Intra-Pastry 루트의 위치 정보들을 저장하고 있다. 이는 새로운 노드의 조인 시 동적 랜드마크(dynamic landmark)로도 사용되는 노드들의 집합이다. 루트노드는 엔트리가 루트노드들로 구성된 Inter-Pastry 라우팅 테이블과 리프노드들로 구성된 Intra-Pastry 라우팅 테이블을 둘 다 가지고 있다. 기존의 Pastry의 라우팅 테이블(그림 1)과 동일한 구조를 갖는다. 리프셋(Leaf-Set)은 자신이 책임지고 있는 Intra-Pastry에 가입된 리프노드들의 정보로 구성되어진다. 리프노드는 기존의 Pastry와 동일하다. Intra-Pastry의 네트워크 토폴로지는 스타나 메쉬(mesh)등 다양한 형태를 고려해 볼 수 있다. 그러나 본 논문에서는 확장성과 일관성을 위해 Pastry를 적용한다. 그림 5는 Rosary 노드의 상태를 보여준다.

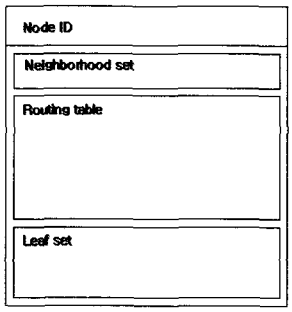


그림 5. Rosary 노드 상태

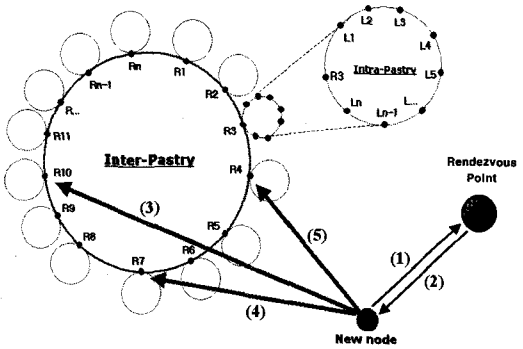


그림 6. Rosary 조인 과정

프락시머티 메트릭은 RTT기반의 인터넷 거리 측정(Internet Distance Estimation) 방법을 사용하며 동적 랜드마크 기법을 이용한다. 동적 랜드마크는 지리적 레이아웃에서 사용되던 고정적인 랜드마크 서버를 두지 않는다. 루트노드가 직접 RTT 측정을 통해 인접한 루트노드들의 집합을 가지며 주기적으로 갱신하여 이 노드들의 집합을 랜드마크 정보로 사용하게 되는 것이다.

Rosary 조인시 잘 알려진(well-known) 캐쉬 서버가 존재한다고 가정했는데 본 논문에서는 RP (Rendezvous Point)라 부른다. Rosary 오버레이 네트워크에 조인하고자 하는 노드 N은 먼저 RP에서 루트노드의 정보를 요청한다. RP로부터 받은 정보로 임의의 루트노드 R1을 선정해서 조인을 하게 된다. N은 프락시머티 메트릭에 따라 R1 까지의 거리를 측정하여 그 결과값이 미리 정해진 조건을 만족할 경우 루트노드와 핸드쉐이킹 (handshaking) 과정을 거치게 된다. 이 과정에서 주고 받은 정보에 따라 노드 N은 R1의 리프노드로 가입하거나 그 Intra-Pastry의 새로운 루트가 될 수 있다. 조건을 만족하지 못할 경우에는 R1으로부터 수신한 이웃셋의 루트노드들에 대해 거리를 측정하여 가장 가까운 루트노드로 다시 조인하게 된다. 이러한 조인과정이 너무 오래 걸리는 것을 방지하기 위해서 TTL 수치(Time To Live number)와 타임아웃 값(timeout value)을 통해 조인작업에 제약을 둔다. 만약 조인이 여의치 않을 경우 Intra-Pastry의 새로운 루트노드로 조인하게 된다. 그림 6은 새로운 노드가 Rosary 네트워크에 조인하기 위한 과정을 간략하게 보여주는 예제이다. (1),(2)는 새로운 노드 N이 Rosary 네트워크에 조인하기 위하여 RP로부터 연결할 지점(R10)에 대한 정보를 얻는다. (3)노드 N은 R10에 대해 RTT를 측정하여 정해진 조건에 만족하면 핸드쉐이킹 과정을 거치고 만족하지 않으면 R10으로부터 이웃셋의 정보를 얻는다. (4),(5)받은 이웃셋에 대해 RTT를 측정하여 가장 가까운 R7으로 동일한 과정을 반복하게 된다. 표 4는 Rosary 조인 알고리즘의 의사코드이다.

표 4. Rosary 조인 알고리즘

```
locateNearNode(new_node)
  boot_nodes = getBootNodes(RP)
  do
    boot_node = selectRandomNode(boot_nodes)
    boot_node_dist = measureDistance(new_node,
boot_node)
    if (IsMeetDistCriterion(boot_node_dist))
      return boot_node
    else
      neighbors = getNeighborhoodSet(boot_node)
      forall neighbor in neighbors
        nb_dist = measureDistance(new_node, neighbor)
        if (IsMeetDistCriterion(nb_dist))
          if (compareDist(near_nb_dist, nb_dist))
            near_neighbor = neighbor
            near_nb_dist = nb_dist
      if (near_negibor != NULL)
        return near_neighbor
  while (!boot_nodes.isEmpty())
  createRootNode(new_node)
```

3.4 컨텐츠 라우팅 알고리즘

사용자로부터 컨텐츠 요청을 받았을 경우 요청을 처리하는 컨텐츠 라우팅 알고리즘(Content Routing Algorithm)은 크게 Intra_Pastry 라우팅 절차와 Inter-Pastry 라우팅 절차로 나뉜다. 표 5는 컨텐츠 라우팅을 위한 절차를 나열한 것이다.

먼저 1단계와 2단계에서 Intra-Pastry의 자신을 포함한 리프노드 중에 요청 컨텐츠가 존재하는가를 찾는 것은 기존 해쉬 기반 기법의 Pastry 라우팅 절차(2장의 표 1)와 동일하다. 1단계 미발견시 상위 Inter-Pastry 라우팅을 수행하게 되는데 이경우 CDN 환경에 적합하게 구성된 Pastry 라우팅 절차 뿐 아니라 질의 및 해쉬 기반의 기법을 적용하게 된다.

표 5. 컨텐츠 라우팅 절차

1단계:	로컬 리프노드(Local Leaf-Node)가 요청한 컨텐츠가 본 서버에 존재할 때 직접 요청 컨텐츠 전송
2단계:	1단계에서 컨텐츠 미발견 시 소속된 Intra-Pastry 라우팅 절차를 수행
3단계:	2단계에서 컨텐츠 미발견 시 Inter-Pastry 라우팅 절차를 수행
4단계:	정해진 RTT 수치, 타임아웃 값 내에서 컨텐츠 미발견시 원본 컨텐츠 서버로 컨텐츠를 요청하여 복제 후 요청 사용자에게 전송

표 6. Inter-Pastry 라우팅 절차

1단계:	이웃셋 내의 모든 루트노드로 쿼리 메시지 전송
2단계:	쿼리를 수신한 루트노드는 Intra-Pastry 라우팅 절차를 수행하여 컨텐츠가 존재하는 경우 응답 메시지 송신. 존재하지 않는 경우 RTT 수, 타임아웃 값 내에서 자신의 이웃셋 내의 모든 루트노드에게 포워딩하며 동시에 Inter-Pastry 상에서 기존의 Pastry 라우팅 절차를 진행
3단계:	2단계에서도 컨텐츠가 존재하지 않거나 제약 조건에 걸렸을 경우 원본 컨텐츠 서버로 컨텐츠를 요청하여 리프노드에 배포하고 리프노드는 사용자에게 전송

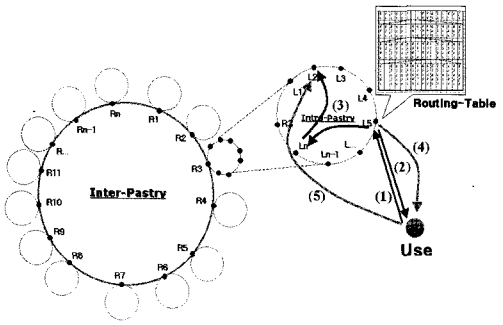


그림 7. Rosary Intra-Pastry 라우팅 과정

Inter-Pastry의 루트노드들은 CDN 환경에서 지역을 담당하는 서버(Regional Edge Server) 역할을 하고 원본 서버로부터 배포되는 각각의 루트노드가 관리하는 Intra-Pastry 내에 최소한 하나의 리프노드에 존재하게 되기 때문에 대부분의 경우에 3단계에서 원하는 컨텐츠를 찾을 수 있다. 3단계 실패의 경우 4단계를 수행한다. 표 6은 Inter-Pastry 라우팅 절차를 나타낸다.

그림 7은 Rosary Intra-Pastry 라우팅 과정을 보여주는 예이다. (1)사용자가 컨텐츠를 서비스 받고자 하는 경우 질의를 받은 리프노드 L5가 (2)컨텐츠를 보유하고 있는 경우에는 컨텐츠를 사용자에게 전송을 하게 되며 (3) 보유하지 못한 경우에는 라우팅 테이블을 통해 컨텐츠가 존재하는 L2의 위치정보를 사용자에게 통지하여 직접 L2로부터 컨텐츠를 전송 받을 수 있게 한다.

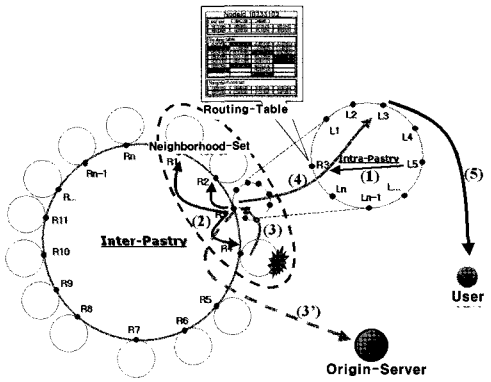


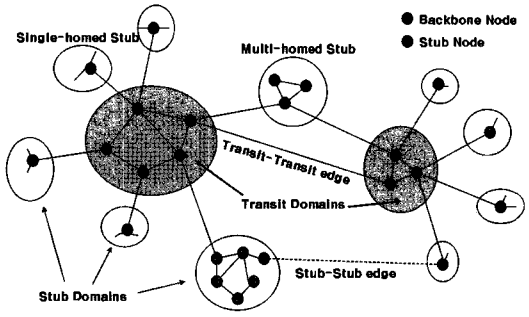
그림 8. Rosary Inter-Pastry 라우팅 과정

그림 8은 Rosary Inter-Pastry 라우팅 과정을 보여주는 예이다. (1)사용자의 요청을 Intra-Pastry 내에서 서비스할 수 없는 경우에 요청을 받은 리프노드 L5는 자신이 소속된 Intra-Pastry의 루트노드 R3에게 컨텐츠 요청을 하게 된다. (2)R3는 자신의 이웃셋의 모든 루트노드 R1, R2, R4에게 질의를 하게 된다. (3)이를 수신한 이웃셋의 루트노드들은 각자 Intra-Pastry 라우팅 과정을 수행하여 질의를 만족하는 리프노드가 존재하면 R3에게 응답을 하여 (4)R3가 자신의 리프노드 L3로 컨텐츠를 복제하여 (5)사용자에게 컨텐츠를 서비스할 수 있게 된다. (3')만약 질의에 만족하는 리프노드가 존재하지 않을 경우엔 원본 서버로 직접 컨텐츠 배포를 요청하게 된다. 라우팅 테이블 갱신 알고리즘(routing table update algorithm)은 Inter-Pastry 와 Intra-Pastry 에서 각각 별도의 기존 Pastry 라우팅 테이블 유지 작업(routing table maintenance task)을 수행한다.

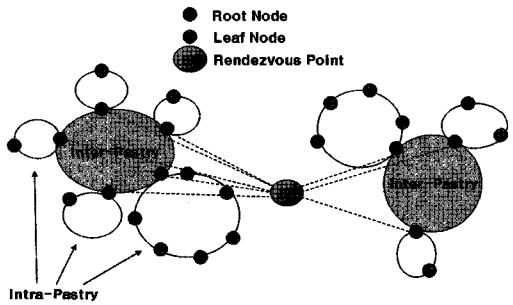
Ⅳ. 성능 평가

4.1 트랜짓-스텁(Transit-Stub) 인터넷 도메인 구조 모델

오늘날의 인터넷은 라우팅 정보를 공유하는 하나의 관리 영역인 라우팅 도메인들이 서로 연결되어 모여 있는것이라 말할 수 있는데 각각의 도메인은 스텁 도메인이나 트랜짓 도메인 둘 중 하나에 속한다. 스텁도메인은 싱글홈과 멀티홈 으로 나뉘어지고



a) 트랜짓-스텁 모델



b) Rosary 모델

그림 9. 인터넷 도메인 구조의 예

멀티홈은 2개 이상의 트랜짓도메인에 소속되어 있다. 트랜짓도메인 내부에는 다수의 백본노드들이 견고하게 연결되어 있으며 각각의 백본노드들은 자신의 스텁도메인을 가진다. 어떤 백본 노드는 직접 또 다른 트랜짓도메인의 백본노드와 직접 연결되기도 하고 다른 트랫짓도메인의 스텁노드 끼리의 연결도 가능하다. 그림 9의 a)는 트랜짓-스텁 인터넷 토폴로지 모델의 예이고 b)는 이것을 시뮬레이션을 위해 Rosary 모델로 바꾸어 본 것이다.[13]

본 논문에서는 트랜짓-스텁 도메인 모델의 개념을 Rosary에 적용하였다. 트랜짓도메인은 Inter-Pastry에 해당하고 스텁도메인은 Intra-Pastry로 볼 수 있다. 트랜짓도메인의 백본노드는 루트노드이고 스텁도메인의 노드는 리프노드로 볼 수 있다. 다음은 네트워크의 필드에 랜덤 하게 뿌려진 노드를 Rosary 의 Inter- Pastry와 Intra-Pastry로 나누고 제어하도록 하는 변수와 전체 노드 수 계산식이다.[13]

N_{inter_Pastry}가 Inter-Pastry의 개수이고 N_{Root}이 Inter-Pastry내의 루트노드의 평균 개수이며 N_{Leaf} 가 Intra-Pastry내의 리프노드의 평균 개수 일때 전체 네트워크의 노드 수 N_{total}은 다음 식 *(1)*과 같다. 식 *(2)*는 루트노드의 평균 개수는 Intra-Pastry의 평균 개수와 동일하다는 의미이다.

$$N_{total} = N_{inter\_Pastry}\, N_{Root}\, N_{Leaf} \quad (1)$$

$$N_{Root} = N_{intra\_Pastry} \quad (2)$$

이런 네트워크의 구조는 Inter-Pastry 내부 비용인 $W(P_{inter_Root} \bullet P_{inter_Root})$와 루트노드와 리프노드간의 비용인 $W(P_{inter\_Root} \bullet P_{intra\_Leaf})$, 리프노드간의 비용인 $W(P_{intra\_Leaf} \bullet P_{intra\_Leaf})$ 값이 도메인 직경(Diameter)인 가장 멀리 떨어진 영역간의 최단 거리값과의 관계에서 다음과 같은 강제 조건을 만족할 때 의도한 트랜짓-스텁 도메인은 효율적으로 만들어지게 된다. 직경 값 D_{Top} 는 Inter-Pastry 사이의 연결거리이고 D_{Inter_Pastry} 는 Inter-Pastry 영역의 최대 직경이다. 또한 D_{Intra_Pastry} 는 Intra-Pastry 영역의 최대 직경 이라고 할 때 다음의 강제조건을 만족하여야 한다. 단, 같은 종류의 간선은 동일한 비용을 갖는 것을 전제로 하며 Inter-Pastry 내의 간선끼리, Intra-Pastry 내의 간선끼리는 동일한 비용을 갖는다. 그러나 두개의 도메인을 연결하는 간선의 경우는 충분히 큰 비용을 산정하여 도메인 내의 노드가 일탈하지 않고 구성원으로 남아 있도록 유도한다.

$$2W(P_{intra\_Leaf} \bullet P_{intra\_Leaf}) > D_{Intra\_Pastry} \quad (3)$$

$$2W(P_{inter\_Root} \bullet P_{inter\_Root}) > D_{Inter\_Pastry} \quad (4)$$

$$2W(P_{inter\_Root} \bullet P_{intra\_Leaf}) > D_{Top}$$
$$W(P_{inter\_Root} \bullet P_{inter\_Root}) + (D_{Top} + 1) D_{Inter\_Pastry} \quad (5)$$

$$2W(P_{intra\_Leaf} \bullet P_{intra\_Leaf}) > 2D_{Intra\_Pastry} + 2W(P_{inter\_Root} \bullet P_{intra\_Leaf}) + D_{Top}\ W(P_{inter\_Root} \bullet P_{inter\_Root}) + D_{Top}\ D_{Inter\_Pastry} \quad (6)$$

$$W(P_{intra\_Leaf} \bullet P_{intra\_Leaf}) \approx 2W(P_{inter\_Root} \bullet P_{intra\_Leaf}) + e \quad (7)$$

위의 식 *(3)*을 만족하게 되면 Inter-Pastry 영역으로의 라우팅 패스보다 Intra-Pastry영역으로의 라우팅 패스가 우선권을 가지게 되고 식 *(4)*를 만족하면 Inter-Pastry간의 Top레벨의 라우팅 패스보다 Inter-Pastry 내부의 라우팅 패스가 우선권을 갖게 된다. 식 *(5)*는 두개의 루트노드가 연결되는 최단의 경로가 유지 될 수 있도록 여분의 간선이 추가로 존재할 수 있음을 의미하며 식*(6)*은 Inter-Pastry 내부의

패스 유지의 효율성을 보장하도록 한다. 식 (7)은 효율성에 따라 Intra-Pastry 내의 리프노드 끼리 직접 연결하는 것을 허용하는 조건이고 e 는 상수이다. 조금은 복잡해 보이는 식을 간단히 정리하면 다음과 같으며 아래 조건을 만족할 경우 메시지 라우팅시 Rosary 내에서 소요되는 비용이 효과적으로 유지되는 네트워크를 구성할 수 있다.

$$W(P_{inter\_Root} \cdot P_{inter\_Root}) := \lceil D_{Inter\_Pastry} / 2 \rceil \quad (8)$$

$$W(P_{inter\_Root} \cdot P_{intra\_Leaf}) := \lceil D_{Top} \; W(P_{inter\_Root} \cdot P_{inter\_Root}) / 2 \rceil + \lceil (D_{Top} + 1) D_{Inter\_Pastry} \rceil \quad (9)$$

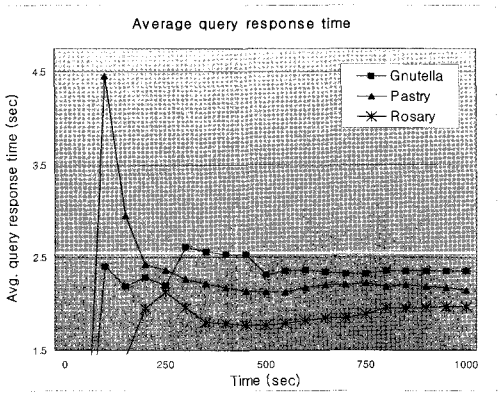
$$W(P_{intra\_Leaf} \cdot P_{intra\_Leaf}) := D_{Intra\_Pastry} + 2W(P_{inter\_Root} \cdot P_{intra\_Leaf}) \quad (10)$$

식 (8)~(10)은 루트노드간의 라우팅 비용의 최소화하는 인터넷상의 논리적인 오버레이 네트워크를 구성하기 위해 Inter-Pastry와 Intra-Pastry의 직경, 루트-루트, 혹은 루트-리프, 리프-리프 사이의 비용을 대입하고 산출된 값을 정수화하는 조건식을 나열하였다.

4.2 시뮬레이션 결과

본 논문에서 제안한 Rosary 오버레이 네트워크의 시뮬레이션을 위해 먼저 GT-ITM(Georgia Tech Internetwork Topology Modes)[13][14] 토폴로지 생성기(topology generator)로 트랜짓-스텁 모델(transit-stub model)을 참조한 Rosary 네트워크 토폴로지를 생성하였다. 생성된 토폴로지는 백본(backbone)으로 100개의 트랜짓-스텁 네트워크 토폴로지를 가지며 각각의 스텁 노드에 랜덤하게 10개에서 20개 사이의 리프노드를 부착하고 이들 리프노드 중 일부에 Rosary 에이전트(peer)를 부착하였다. 이 토폴로지를 사용하여 언더라잉 네트워크 시뮬레이터(underlying network simulator)인 NS-2(Network Simulator Version 2)[15]상에서 시뮬레이션을 수행하였다.

Rosary 에이전트는 그누텔라심(GnutellaSim)[16]에서 사용하는 P2P 시뮬레이터 프레임웍(P2P simulator framework)을 기반으로 구현되었다. 이 프레임웍은 다양한 P2P 프로토콜을 네트워크 시뮬레이터(예: NS-2, GTNets 등) 상에서 더 쉽게 시뮬레이션 할 수 있는 환경을 제공하며 그누텔라심을 이용해서 그누텔라 프로토콜을 시뮬레이션할 수 있다. 피어 행위 모델(peer behavior model)은 참고문헌[17]에서 사용된 피어 행위 모델을 적용하였다. 시뮬레이션은 동일한 환경 하에서 Gnutella, Pastry 그리고 본 논문에서 제안한 Rosary 시스템을 비교하여 수행되었다. CDN 환경에선 사용자가 컨텐츠를 요청하였을 때의 응답시간이 매우 중요한 사항이며 서비스의 품질과도 밀접한 관련이 있기 때문에 평균 질의 응답 시간에 대한 시뮬레이션 결과를 먼저 분석하였다.

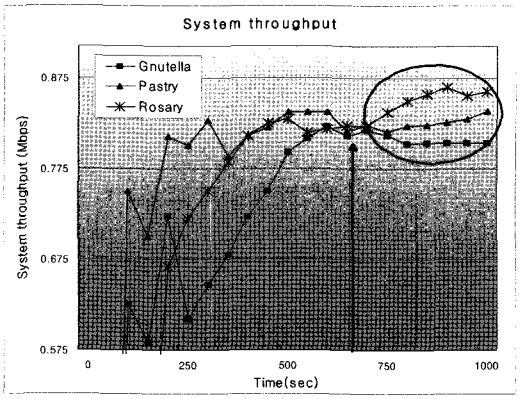


시스템	평균 질의 응답 시간(sec)
Gnutella	2.355910
Pastry	2.142328
Rosary	1.965853

그림 10. 평균 질의 응답 시간

그림 10은 1000초 동안 평균 질의 응답 시간을 비교한 것이다. 본 논문에서 제안한 Rosary 시스템이 가장 짧은 응답시간을 보이고 있는데 이러한 결과는 원하는 컨텐츠를 검색하기 위한 질의가 Rosary 오버레이 네트워크 상에서 효율적으로 수행되고 있어 Rosary 시스템이 다른 시스템에 비해 가장 CDN 환경에 적합하게 설계되어졌다는 것을 보여준다. 컨텐츠를 검색하기 위해 질의를 수행하였을 경우 실제로 만족하는 질의의 응답 유무(시스템 처리율)에 대한 시뮬레이션 결과를 분석하였다.

그림 11은 요청된 전체 질의에 대해 만족한 응답을 얻은 쿼리와 1000초 동안 시스템 처리율을 나타낸다. Rosary는 그래프의 전반부에서 볼 수 있듯이 Pastry와 비교해서 오버레이 네트워크를 운영하기 위한 구성시간이 좀 더 길다. 그러나 초기 네트워크 구성 단계를 지나 600초 이후에 안정한 상태로 동작하기 시작하면서 Rosary 시스템의 처리율이 가장 좋은 상태를 꾸준히 유지하는 것으로 나타났다.

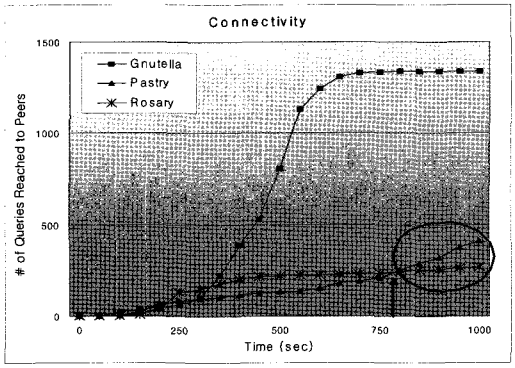


시스템	시스템 처리율 (Mbps)
Gnutella	0.8032787
Pastry	0.8383838
Rosary	0.8593750

그림 11. 시스템 처리율

성능분석의 기준으로 평균 질의 응답시간과 시스템 처리율, 피어로 전달된 질의의 수를 측정하였다. 전체 네트워크에서의 트래픽의 전송량 및 빈도, 집중화 정도에 따라 다양한 결과가 산출된다. 시스템 처리율이란 Gnutella, Pastry, Rosary 로 구현된 시스템이 똑 같은 질의를 요구받을 경우 초당 처리하게 되는 데이터의 양을 의미하며 현 시뮬레이션 모델에서의 1ksec 시뮬레이션 지속시간을 증가시키면 지속되는 시간만큼 큰 폭으로 증대된다. 이는 일반적인 P2P 네트워크의 구성 후 지속시간을 고려해 볼 때 초기 망 구성에 소요되는 시간을 제외한 경우 훨씬 높은 처리율을 보이는 것이고 전체 시간을 놓고 볼 때도 Gnutella와 비교해서 초기 구성 시간의 낮은 처리율을 보상하여 망 운영 측면에서 분명한 성능의 향상이라 볼 수 있다. 다음으로 컨텐츠를 검색하기 위해 실제로 수행된 질의의 수에 대한 시뮬레이션 결과를 분석하였다.

그림 12는 1000초 동안 피어로 전달된 질의 수를 나타낸다. 그래프의 전반부에서 볼 수 있듯이 Rosary는 Pastry와 비슷하거나 약간 낮은 숫치를 보였다. 이는 오버레이 네트워크를 운영하기 위한 구성시간이 Rosary의 경우 더 소요되기 때문이고 초기 네트워크 구성 단계를 지나 700초 이후에 네트워크가 안정된 상태로 진입하면서 Rosary 시스템의 도착 질의 수가 가장 좋은 상태를 꾸준히 유지하는 것으로 나타났다. Rosary는 세미 해쉬 기법의 적용으로 과도한 트래픽을 유발하지 않고 좋은 질의 응답 성공률을 가진 것을 알 수 있다.



시스템	피어로 전달된 질의 수(개)
Gnutella	1340
Pastry	414
Rosary	267

그림 12. 피어로 전달된 질의 수

다음 표 7은 지금까지의 시뮬레이션 결과를 바탕으로 분석 항목 별로 Rosary 시스템과 Gunetella, Rosary 시스템과 Pastry를 비교하여 Rosary의 개선률을 %로 나타낸 것이다.

표 7. Rosary 개선률 (vs. Gnutella, Pastry)

분석 항목	Gnutella	Pastry
평균 질의 응답 시간	16.56%	8.24%
시스템 처리율	6.98%	2.50%
피어로 전달된 질의 수	80.07%	35.51%

V. 결 론

이 논문에서는 CDN 시스템에 적합한 새로운 Rosary 오버레이 네트워크를 제안하였다. Rosary는 기존의 P2P 시스템 중 Pastry의 기능을 CDN 환경에 맞게 수정, 확장한 오버레이 네트워크 시스템이다. 기존의 Pastry는 특정 노드에 부하가 집중되는 현상과 실제로는 인접한 노드가 라우팅의 대상에서 제외될 수 있다는 점에서 CDN 시스템으로 적합하지 않다. Rosary는 Pastry를 Inter-Pastry와 Intra-Pastry로 나누어 CDN 환경에 맞게 확장하였고 애플리케이션-레벨 멀티캐스팅(Applicatio-Level Multicasting)을 가능하도록 했으며 세미 해쉬 기법(Semi Hash Scheme)을 적용하였다. Pastry의 프락시머티 이웃 선택(Proximity Neighbor Selection) 기술에 지리적 레이아웃(Geographic Layout) 기술을 적용, Rosary 노드가 직접 RTT를 측정하여 랜드마크 서버의 역할을 하게하였다. 그래서 완전한 셀프 오거

나이징(Self-Organizing)의 특성을 유지하면서도 Rosary 공간의 Inter-Pastry 엔트리들이 중요한 컨텐츠를 적절히 분산하여 갖도록하여 핫스팟을 유발하는 불균형을 개선하였다. 성능분석을 위해 실제 인터넷 망의 모델을 이용하여 환경을 구성하였고 NS-2를 통하여 성능의 향상을 검증하였다. 그 결과 Rosary 시스템은 각각 Gnutella, Pastry와 비교하여 가장 낮은 질의 횟수와 가장 짧은 평균 응답 시간을 보였으며 시스템 전송에서도 상대적으로 좋은 처리율을 유지하였다. 좀 더 크고 분산된 대규모의 P2P 시스템일수록, 또 P2P 네트워크의 유지시간이 길어질수록 더 좋은 성능을 보이는 특징이 갖고 있어 앞으로 대규모 WAN에서의 IP를 통한 혼잡한 대규모 컨텐츠 배포에 더욱더 유리한 기법임을 알 수 있다. 이러한 결과는 다른 시스템과 비교하여 오버레이 네트워크 상에서 컨텐츠 검색을 위해 쿼리와 응답, 제어 메시지의 송수신등 과도하고 쓸모없는 트래픽을 유발하지 않고서도 원하는 컨텐츠의 검색 및 배포 작업을 효율적으로 수행할 수 있음을 보여주는 것이다.

전반적인 성능의 향상에도 불구하고 Rosary 시스템은 다른 시스템에 비하여 초기 오버레이 시스템 형성시에 좀더 많은 시간을 필요로 하는 문제점을 지니고 있다. 앞으로 개선해야할 과제로 남겨져 있으며 본 연구를 바탕으로 Rosary 시스템과 다양한 네트워크 토폴로지(매쉬형, 스타형)와의 통합모델을 연구할 수 있다. 또한 최근 관심이 증대되고 있는 USN(Ubiquitous Sensor Network)이나 애드혹(Ad-hoc), WSN(Wireless Sensor Network)의 물리적 라우팅 관리기법으로 적용하도록 연구할 수 있을 것이다.

참 고 문 헌

[1] Bjurefors, F., Larzon, L.A., and Gold, R., "Performance of Pastry in a heterogeneous system," Peer-to-Peer Computing Proceedings, Fourth International Conference, 25-27 Aug. 2004.

[2] Swart, G., "Spreading the load using consistent hashing: a preliminary report," Models and Tools for Parallel Computing on Heterogeneous Networks, Third International Symposium, 5-7 July 2004.

[3] Gnutella Website. http://gnutella.wego.com

[4] Freenet Website. http://freenet.sourceforge.net

[5] Napster Website. http://www.napster.com

[6] Morpheus Website. http://www.musiccity.com

[7] Kazaa Website. http://www.kazaa.com

[8] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content-addressable network," ACM SIGCOMM, San Diego, CA, Aug. 2001.

[9] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," ACM SIGCOMM, San Diego, CA, Aug. 2001.

[10] A. Rowstron and P. Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems," 18th IFIP/ACM Int. Conf. Distributed Systems Platforms (Middleware 2001), Heidelberg, Germany, Nov. 2001.

[11] B. Zhao, J. Kubiatowicz, and A. Joseph, "Tapestry: An infrastructure for fault-tolerant wide-area location and routing," Univ. California, Berkeley, CA, TR UCB/CSD-01-1141, Apr. 2001.

[12] Miguel Castro, Peter Druschel, Y. Charlie Hu and Antony Rowstron, "Topology-Aware Routing in Structured peer-to-peer overlay networks," Microsoft Research Technical Report MSR-TR-2002-82.

[13] E. Zegura, K. Calvert, and S. Bhattacharjee, "How to model an internetwork," INFOCOM 96, 1996.

[14] GT-ITM http://www.cc.gatech.edu/projects/gtitm

[15] NS-2 http://www.isi.edu/nsnam/ns

[16] Q. He, M. Ammar, G. Riley, H. Raj, and R. Fujimoto, "Mapping Peer Behavior to Packet-level Details: A Framework for Packet-level Simulation of Peer-to-Peer Systems", MASCOTS, 2003.

[17] Z. Ge, D. R. Fiegueiredo, S. Jaiswal, J. Kurose, and D. Towsley, "Modeling peer-peer file sharing systems," INFOCOM, 2003.

신 수 영 (Soo-Young Shin) 정회원

1998년 한국방송통신대학교 교육학사

2002 년 덕성여자대학교 정보 통신 이학석사

2004년~현재 국민대학교 비즈니스 IT 박사과정

2001년 8월~현재 신흥대학교 강사

2003년 3월~현재 덕성여자대학교 강사

2004년 8월~현재 국민대학교 강사

<관심분야> MAC scheduling, WLAN, WPAN, WSN, USN

남궁 정 일 (Jung-Il Namgoong) 정회원

1998년도 인천대학교 기계공학사

2005년도 국민대학교 비즈니스 IT 정보통신 이학석사

<관심분야> CDN, Network management, P2P network

박 수 현 (Soo-Hyun Park) 정회원

1988년 고려대학교 컴퓨터학과 이학사

1990년 고려대학교 전산학 이학석사

1998년 고려대학교 컴퓨터학과 이학박사

1990년~1999년 (주)LG전자 중앙연구소 선임연구원

1999년~2001년 동의대학교 공과대학 소프트웨어공학과 교수

2002년~현재 국민대학교 비즈니스IT학부 교수

<관심분야> 이동통신 시스템, Active Network, 유비쿼터스 네트워크