구조화된 토폴러지-인식 P2P 오버레이 네트워크에서의 콘텐츠 라우팅 메커니즘

남궁정일, 신수영, 박수현*

Contents Routing Mechanism for Topology-Aware Structured P2P Overlay Network

Jeong-Il Namgung, Soo-Young Shin, Soo-Hyun Park

Abstract

최근의 제 2 세대 P2P 시스템은 확장성 있고 장애에 강한 분산 해쉬 테이블을 이용하여 대규모 P2P 애플리케이션을 위한 셀프-오거나이징 (Self-Organizing)기술을 제공한다. 하지만 기존의 Pastry의 경우 해쉬 기반 기법을 적용하였기 때문에 사용자들의 요청이 빈번한 콘텐츠의 경우 한 노드로 네트워크의 전송 부하가 집중되는 현상이 발생할 수 있어 CDN 시스템에 적합하지 않았다. 본 논문에서는 이러한 문제점을 해결하기 위하여 CDN(Content Delivery Network) 환경에 적합하게 확장한 Rosary 오버레이 네트워크를 제안한다. 본 논문에서 제안하는 Rosary는 Pastry를 Inter-Pastry와 Intra-Pastry로 나누어 CDN 환경에 맞게 확장하였고 애플리케이션-레벨 멀티캐스팅(Applicatio-Level Multicasting)을 가능하도록 했으며 세미 해쉬 스킴(Semi Hash Scheme)을 적용함으로서 CDN 환경에 맞게 수정하고 확장하였다.

Key Words: P2P overlay networks, Content Delivery Network(CDN), Rosary

*국민대학교 비즈니스IT학부 비즈니스 정보통신연구실

1. 서 론

그누텔라(Gnutella)[1], 프리넷(Freenet)[2], 냅스터(Napster)[3], 모피어스(Morpheus) [4], 카자(Kazaa)[5] 등과 같은 제 1 세대 P2P 파일 공유 시스템으로부터 최근의 제 2 세대 P2P 시스템인 CAN[6], Chord[7], Pastry [8], Tapestry[9] 와 같은 분산 P2P 시스템과 그 시스템을 활용한 애플리케이션들이 활발히 연구되고 있다. 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]. 논문에서는 Pastry를 기반으로 CDN(Content Delivery Network) 환경에 적합하게 확장한 Rosary 오버레이 네트워크를 제안한다. 제안한 오버레이 네트워크의 토폴로지 구성이 천주교의 묵주의 연결체 모양을 하고 있기에 Rosary라는 이름을 붙였다. 기존의 Pastry의 경우 해쉬 기반 기법을 적용하였기 때문에 사용자들의 요청이 빈번한 콘텐츠의 경우 한 노드로 네트워크의 전송 부하가 집중되는 현상이 발생할 수 있어 CDN 시스템에 적합하지 않았다. 본 논문에서 제안하는 Rosary는 이러한 점을 극복하기 위하여 기존의 Pastry를 Inter-Pastry와 Intra-Pastry로 구분하여 그 토폴로지를 좀 더 CDN 환경에 맞게 수정하고 확장하였다. 즉 애플리케이션-레벨 멀티캐스팅 (Application- Level Multicasting)을 위해 그룹 개념인 Intra-Pastry를 추가하여 완전 해쉬 스킴(Fully Hash Scheme)이 아닌 세미 해쉬 스킴(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.

2. Rosary

2.1 Inter-Pastry / Intra-Pastry

본 논문에서 제안한 Rosary는 기존의 Pastry 기법을 기반으로 하여 CDN 시스템에 적합하게 확장한 구조화된 오버레이 네트워크이다. 기존의 Pastry의 경우 해쉬 기반의 스킴을 적용하였기 때문에 콘텐츠를 특정 Pastry 노드 한 곳에만 배포를 하게 된다. 이 방법은 사용자들의 요청이 빈번한 자료의 경우에 부하가 한 노드로 집중되는 현상이 발생할 수 있기 때문에 CDN 시스템에 적합하지 않다. Pastry는 키에 대한 노드ID 프리픽스(prefix)를 기반으로 라우팅을 하기 때문에 키와 다른 노드ID를 가지면서 현재 노드와 인접해 있는 노드들이 그 대상에서 제외되어 최악의 경우 실제로 인접한 노드의 절반 이상이 반영되지 못하는 경우가 발생할 수 있다. Rosary는 이러한 점을 극복하기 위하여 기존의 Pastry를 Inter-Pastry와 Intra-Pastry로 구분하여 그 토폴로지를 좀 더 CDN 환경에 맞게 확장하였다. .

Inter-Pastry는 루트(root) 노드들로 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 서버(proxy server)의 역할도 수행한다. 루트노드는 콘텐츠가 존재하는 원본 서버(origin server)와의 통신을 통해 자신이 관리하는 Intra-Pastry에 가입된 리프노드(leaf node)로 콘텐츠 배포를 담당하며 해당 리프노드가 콘텐츠를 효율적으로 전송 할 수 있도록 네트워크 부하를 분산시키는 역할을 수행한다. 이렇게 하여 원본 서버로의 과도한 부하의 집중을 방지할 수 있다.

Intra-Pastry는 리프노드(Leaf-Node)들로 구성되어지며 하나의 루트노드에 의하여 관리되어진다. 리프노드는 각 지역의 경계 서버(Edge Server)로서 그 지역과 인접한 유저들의 콘텐츠 요청에 대해 콘텐츠를 직접 전송하는 역할을 수행하며 루트노드의 장애 등으로 루트노드가 존

재하지 않게 되면 같은 Intra-Pastry에 가입된 모든 리프노드들 중에서 가장 성능이 좋은 노드가 대체 서버로서의 역할을 수행하게 된다. 그림 1 제안된 Rosary 토폴로지를 도식화 한 것이다.

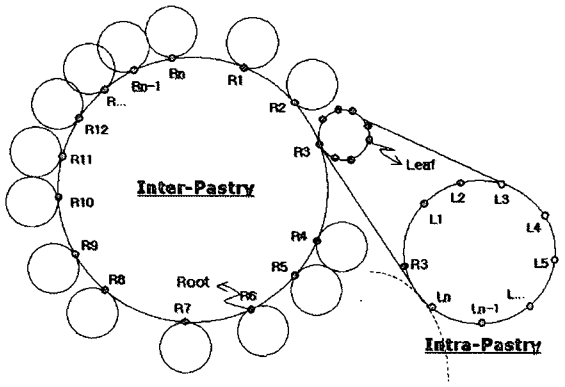


그림 1 Rosary 토폴로지

콘텐츠의 배포 및 전송의 경우 원본 서버로부터 콘텐츠의 배포 요청이 루트노드에 도착하면 각각의 루트노드는 자신에게 조인되어 있는 Intra-Pastry의 모든 리프노드에게 콘텐츠를 배포한다. 또한 Intra-Pastry 내에서 자신과 인접한 루트노드에게도 콘텐츠를 배포하게 된다. 만약 사용자가 요청한 콘텐츠가 Intra-Pastry내에 존재하지 않으면 루트노드는 인접한 다른 루트노드에게 콘텐츠의 배포를 요청하며 실패 시 원본 서버에게 콘텐츠를 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 2는 Rosary의 노드ID 구조를 나타낸다.

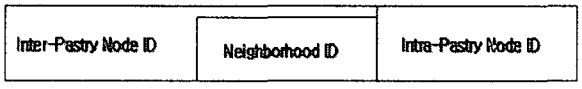


그림 2 Rosary 노드ID 구조

2.2 콘텐츠 라우팅 알고리즘

사용자로부터 콘텐츠 요청을 받았을 경우 요청을 처리하는 콘텐츠 라우팅 알고리즘(Content Routing Algorithm)은 크게 Intra_Pastry 라우팅 절차와 Inter-Pastry 라우팅 절차로 나뉜다. 표 1은 콘텐츠 라우팅을 위한 절차를 나열한 것이다.

먼저 1단계와 2단계에서 Intra-Pastry의 자신을 포함한 리프노드 중에 요청 콘텐츠가 존재하는가를 찾는 것은 기존 해쉬 기반 스킴의

표 1 콘텐츠 라우팅 절차

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

Pastry 라우팅 절차와 동일하다. 1단계 미발견 시 상위 Inter-Pastry 라우팅을 수행하게 되는데 이 경우 CDN 환경에 적합하게 구성된 Pastry 라우팅 절차 뿐 아니라 질의 및 해쉬 기반의 스킴을 적용하게 된다. 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 콘텐츠를 찾을 수 있다. 3단계 실패의 경우 4단계를 수행한다. 표 2는 Inter-Pastry 라우팅 절차를 나타낸다.

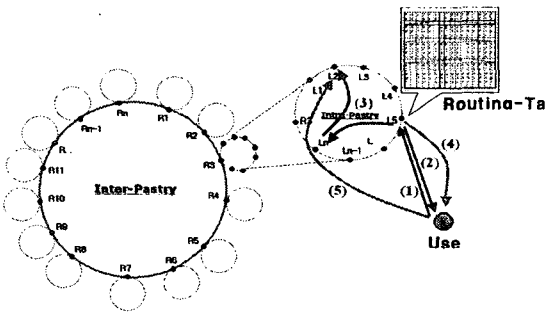


그림 3 Rosary Intra-Pastry 라우팅 과정

표 2 Inter-Pastry 라우팅 절차

1단계:	이웃셋 내의 모든 루트노드로 쿼리 메시지 전송
2단계:	쿼리를 수신한 루트노드는 Intra-Pastry 라우팅 절차를 수행하여 콘텐츠가 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 콘텐츠가 존재하지 않거나 제약 조건에 걸렸을 경우 원본 콘텐츠 서버로 콘텐츠를 요청하여 리프노드에 배포하고 리프노드는 사용자에게 전송

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

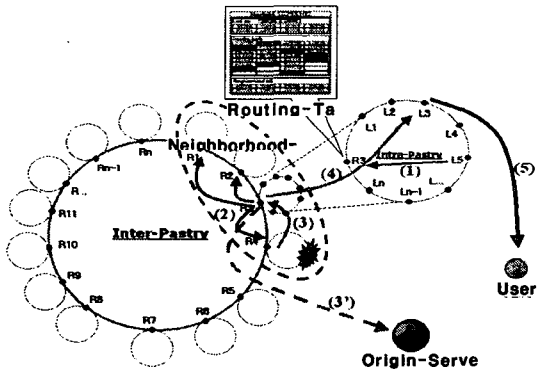


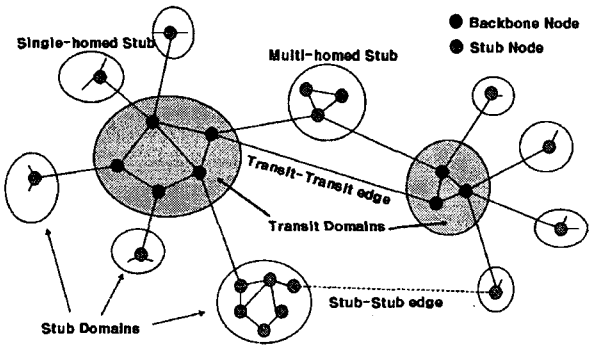
그림 4 Rosary Inter-Pastry 라우팅 과정

그림 4는 Rosary Inter-Pastry 라우팅 과정을 보여주는 예이다. (1)사용자의 요청을 Intra-Pastry 내에서 서비스할 수 없는 경우에 요청을 받은 리프노드 L5는 자신이 소속된 Intra-Pastry 의 루트노드 R3에게 콘텐츠 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 콘텐츠를 복제하여 (5)사용자에게 콘텐츠를 서비스할 수 있게 된다. (3')만약 질의에 만족하는 리프노드가 존재하지 않을 경우엔 원본 서버로 직접 콘텐츠 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.

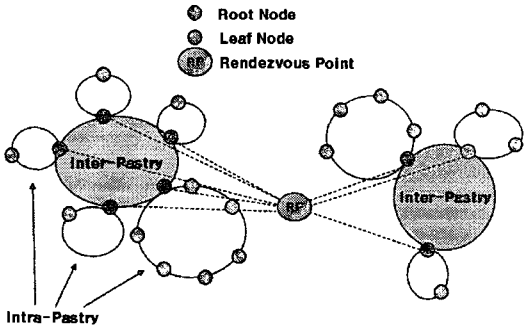
3. 성능평가

3.1 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 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 5의 a)는 트랜짓-스텁 인터넷 토폴로지 모델의 예이고 b)는 이것을 시뮬레이션을 위해 Rosary 모델로 바꾸어 본 것이다. [13]



a) 트랜짓-스텁 모델



b) Rosary 모델

그림 5 인터넷 도메인 구조의 예

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. [12]
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} 는 다음 식 (1)과 같고 식(2)는 루트노드의 평균 개수는 Intra-Pastry의 평균개수와 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 $W(P_{inter\_Root} \cdot P_{inter\_Root})$ 와 루트노드와 리프노드간의 비용인 $W(P_{inter\_Root} \cdot P_{intra\_Leaf})$, 리프노드간의 비용인 $W(P_{intra\_Leaf} \cdot P_{intra\_Leaf})$ 값이 도메인 직경(Diameter)인 가장 멀리 떨어진 영역간의 최단 거리값과의 관계에서 다음과 같은 강제조건을 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.

$$2W(P_{intra\_Leaf} \cdot P_{intra\_Leaf}) > D_{Intra\_Pastry} \quad (3)$$
$$2W(P_{inter\_Root} \cdot P_{inter\_Root}) > D_{Inter\_Pastry} \quad (4)$$

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

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

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

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.

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

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

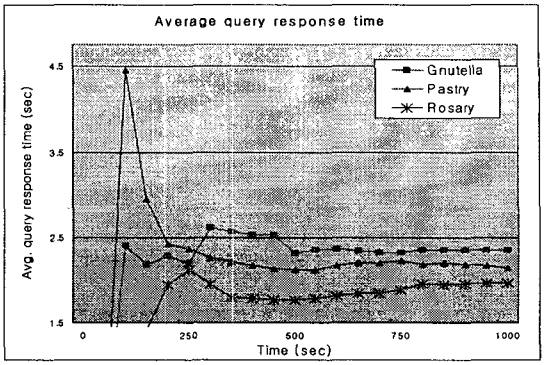
$W(P_{intra\_Leaf} \cdot P_{intra\_Leaf}) :=$
$D_{Intra\_Pastry} + 2W(P_{inter\_Root} \cdot P_{intra\_Leaf})$ (10)

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

3.2 시뮬레이션

본 논문에서 제안한 Rosary 오버레이 네트워크의 시뮬레이션을 위해 먼저 GT-ITM(Georgia Tech Internetwork Topology Modes)[7][13] 토폴로지 생성기(topology generator)로 트랜짓-스텁 모델(transit-stub model)을 참조한 Rosary 네트워크 토폴로지를 생성하였다. 생성된 토폴로지를 사용하여 언더라잉 네트워크 시뮬레이터(underlying network simulator)인 NS-2(Network Simulator Version 2)[6]상에서 시뮬레이션을 수행하였다.

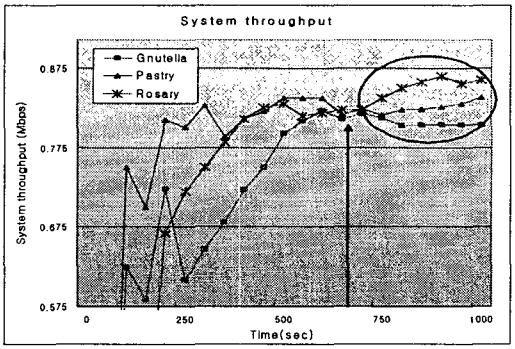
Rosary 에이전트는 그누텔라심(GnutellaSim)[13]에서 사용하는 P2P 시뮬레이터 프레임웍(P2P simulator framework)을 기반으로 구현되었다. 이 프레임웍은 다양한 P2P 프로토콜을 네트워크 시뮬레이터(예: NS-2, GTNets 등) 상에서 더 쉽게 시뮬레이션 할 수 있는 환경을 제공하며 그누텔라심을 이용해서 그누텔라 프로토콜을 시뮬레이션할 수 있다. 피어 행위 모델(peer behavior model)은 [14]에서 사용된 피어 행위 모델을 적용하였다. 시뮬레이션은 동일한 환경 하에서 Gnutella, Pastry 그리고 본 논문에서 제안한 Rosary 시스템을 비교하여 수행되었다. CDN 환경에선 사용자가 콘텐츠를 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 6 평균 질의 응답 시간

그림 6은 1000초 동안 평균 질의 응답 시간을 비교한 것이다. 본 논문에서 제안한 Rosary 시스템이 가장 짧은 응답시간을 보이고 있는데 이러한 결과는 원하는 콘텐츠를 검색하기 위한 질의가 Rosary 오버레이 네트워크 상에서 효율적으로 수행되고 있어 Rosary 시스템이 다른 시스템에 비해 가장 CDN 환경에 적합하게 설계되어 졌다는 것을 보여준다.



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

그림 7 시스템 처리율

그림 7은 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. 일반적인 P2P 네트워크의 지속시간을 감안할 때 결과적으로 Rosary 시스템이 가장 좋은 시스템 처리율을 보인다고 할 수 있다.

4. 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 스킴(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 오거나이징(Self-Organizing)의 특성을 유지하면서도 Rosary 공간의 Inter-Pastry 엔트리들이 중요한 콘텐츠를 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 콘텐츠 배포에 더욱더 유리한 기법임을 알 수 있다. 이러한 결과는 다른 시스템과 비교하여 오버레이 네트워크 상에서 콘텐츠 검색을 위해 쿼리와 응답, 제어 메시지의 송수신등 과도하고 쓸모없는 트래픽을 유발하지 않고서도 원하는 콘텐츠의 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 지속적으로 연구하여 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) 네트워크의 물리적 라우팅 관리기법으로 적용하도록 연구할 수 있을 것이다.

[참고 문헌]

[1] Gnutella Website. [Online]. *http://gnutella.wego.com*

[2] Freenet Website. [Online]. *http://freenet.sourceforge.net*

[3] Napster Website. [Online]. *http://www.napster.com*

[4] Morpheus Website. [Online]. *http://www.musiccity.com*

[5] Kazaa Website. [Online]. *http://www.kazaa.com*

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

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

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

[9] 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.

[10] A. Rowstron and P. Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems," presented at the 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] E. Zegura, K. Calvert, and S. Bhattacharjee, "How to model an internetwork," in INFOCOM96, 1996.

[13] 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", in MASCOTS, 2003.

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