논문 2011-5-29

해시 테이블 기반 분산형 CDN 구조 및 서버 선택 방안

Architecture and Server Selection for DHT-based Distributed CDN

정종해, 오건영, 이남경, 윤장우, 이현우, 류원, 이성창*

Jong-Hae Jung, Gun-Young Oh, Nam-Kyung Lee, Chang-Woo Yoon, Hyun-Woo Lee, Won Ryu, Sung-Chang Lee

요 약 중앙집중형 CDN에서는 사용자의 요청에 대해 서비스 망의 노드가 가용한 소스 노드들 중에서 하나를 선택하여 사용자에게 알려준다. 본 논문에서는, 사용자 요청이 오버레이 망에서 DHT(distributed hash table)를 이용한 P2P 알고리듬에 의해 컨텐츠 소스들에게 전달되고, 이 소스들로부터 받은 응답들을 바탕으로 사용자가 실시간 사용자 중심의 선택을 하는 분산형 CDN 구조 및 동작 알고리듬을 제안한다. 이를 위해 분산형 CDN 구조와 함께 수정된 Pastry 방식을 제안하고, 이를 이용한 컨텐츠의 등록, 검색, 선택을 방안을 기술 하였다. 제안한 CDN 구조는 분산형 구조가 갖는 부하 분산, 트래픽 분산, 확장성, 자가 구성 및 자가 회복에 의한 장애에 대한 관용성 등의 장점도 가진다. 다양한 시뮬레이션을 통하여 제안한 방안의 유효성을 보였으며, 제안한 방안의 여러 가지 변형 방안에 대한 성능을 비교, 검토하였다.

Abstract In centralize CDN systems, the content server selection is performed by service node for every user request, and the selected node is notified to the user. In this paper, we present distributed CDN architecture and algorithm in which the request from a user is delivered to the content source by a P2P algorithm utilizing DHT(distributed hash table) through the overlay network and the user selects one of the source nodes based on real-time user-centric criteria. For this purpose, we propose a modified Pastry algorithm for contents registration, search and selection, in addition to the distributed architecture. The proposed architecture has the advantages of load balancing, traffic balancing, scalability, fault-tolerance due to the self-configuration, self-healing attributes of distributed architecture. Various simulation shows the feasibility of the proposed architecture and algorithm, and the performance is compared and discussed for the variations of the proposed scheme.

Key Words : CDN, Server selection, DHT, P-to-P

Ⅰ. 서 론

최근 유비쿼터스 통신의 다양한 기술이 혁신적으로 발전하면서 노드 간, 서비스 간, 이종망간의 연동과 통합이 가속화 되고 있다[1, 2]. 또한, UCC 및 VOD와 같은 대용량 스트리밍 서비스가 이슈화 되면서, 단기간에 급속하게 집중되는 트래픽을 어떻게 하면 안전하고 빠르게 처리할 수 있는지에 대한 관심이 증대되었다. 이러한 배경을 기반으로 CDN(Content Delivery Network)이 등장하게 되었다.

CDN은 콘텐츠 제공업자(CP)의 웹 서버에 집중되어 있는 콘텐츠 용량이 크거나 사용자의 요구가 빈번한 콘텐츠들을 ISP측에 설치한 CDN 서버에 미리 저장, CDN 서버로부터 최적의 경로로 사용자에게 콘텐츠를 전달하는 기술이다. 동영상이나 음악 스트리밍, 파일 다운로드 등 대용량 파일 전송 시 이용자가 몰려 전송 속도가 떨어질 때, 네트워크 주요 지점에 전용 서버를 설치해 두고

*정회원, 한국항공대학교 항공전자 및 정보통신공학부
접수일자 2011.8.31, 수정완료 2011.9.30
게재확정일자 2011.10.14

해당 콘텐츠를 미리 저장한 후 이용자가 몰릴 때 가까운 곳의 서버가 이를 전송하게 된다. CDN은 인터넷에서 유용하게 사용되고 있으며 나날이 그 중요성이 커지고 있다. 이는 앞으로 진정한 유비쿼터스 통신의 실현에서 다양한 발전으로의 교각역할을 해 낼 것으로 보여 진다[2].

일반적인 인터넷 구조에서는 콘텐츠는 CP의 서버로부터 ISP들의 백본 네트워크, IX(Internet exchange), 가입자망 등 복잡한 경로를 거쳐 사용자에게 전달되지만 CDN 서비스는 콘텐츠 전송에 병목현상을 일으키는 구간을 경유하지 않고, 인터넷 사용자의 가장 가까운 지점에서 콘텐츠를 전송하기 때문에 대용량 콘텐츠나 사용자가 일시에 폭증하는 경우에도 빠르고 안정적인 서비스가 가능하다. 용량이 크거나 사용자가 몰리는 콘텐츠를 인터넷 서비스 제공업자 측에 설치한 CDN 서버에 미리 저장해 최적의 경로로 사용자에게 전달할 수 있고, 특정 ISP의 장애 시 에도 타 ISP에 설치된 CDN 서버를 통해 서비스가 가능하다.

현재까지의 CDN들은 대부분 계층적 중앙집중 구조를 가지고, 관련 연구들도 이러한 구조를 가정하고 있다. 중앙집중형 CDN 구조에서는, 사용자의 매 서비스 요청에 대해서 서비스 제공자 측의 특정 노드가 컨텐츠 소스 선택 기능을 가지며, 사용자가 요청한 컨텐츠를 서비스할 수 있는 소스 노드들 중에서 소스 노드 선택 방안을 사용하여 사용자와 컨텐츠 소스 간의 선택 기준에 의거해서 소스 서버를 선택하여 사용자에게 알려준다.

중앙집중 구조는 나름대로의 장점도 있지만, 서비스 요청 및 제공을 위한 각종 부하가 집중되고, 또한 트래픽도 망의 각 링크에 병목을 야기하는 경우가 있으며, 확장성, 장애에 대한 관용성 보장 등의 측면에서 분산형 구조에 비해 불리할 수 있다.

이것에 비해 본 논문에서는 분산형 서버 선택 구조를 제안하며, 사용자의 요청은 분산형 해쉬 테이블(DHT)을 이용한 P2P 알고리듬이 동작되는 오버레이 네트워크 상에서 컨텐츠의 등록, 검색, 선택 등이 수행된다.

P2P 방식에 대해서는 상당기간 많은 연구가 이루어졌고, 다양한 형태의 많은 방식이 제안되었다. 중앙집중형(cnetralized)의 P2P는 앞서 기술한 중앙집중형 CDN가 유사한 속성을 가질 것이고, Unstructured 형태의 P2P들은 탐색 과정에서 기본적으로 Flooding 방식에 의존하므로 트래픽을 무분별하게 발생 시킬 수 있어서, 본 논문의 분산 동작 방식에는 부적합하다. 동작 시의 트래픽 발생을 보다 절제하면서도 효과적으로 동작하는 P2P들이 많이 제안되었는데, [19]에서는 원형 주소 공간에 각각의 노드와 데이터의 키 값을 할당한 뒤 DHT(distributed hash table) 알고리듬을 이용하여 원하는 노드 정보를 얻는 Chord를 제안하였다. 이러한 DHT 알고리듬을 통해 원활하게 피어의 참여와 이탈을 관리할 수 있고, 전체 라우팅 정보의 일관성을 유지하게 되며 DHT가 적용된 핑거 테이블을 통해 라우팅 횟수를 줄어들게 할 수 있다. [18]에서는 d-차원의 데카르트 좌표 공간을 가상의 논리적 주소로 사용하는 방식인 CAN을 제안하였다. 이를 통해 장애에 강하며 확장성이 있고 자가 구성이 가능한 P2P 네트워크를 구성하였다. [20]에서는 DHT를 적용하여 서로 다른 동적 P2P 시스템에서의 부하 분산을 위한 알고리듬을 제안하였고, 이를 통해 급격한 피어들의 참여와 이탈에 따른 높은 부하를 고르게 분산 시켰다.

이러한 DHT를 이용한 P2P 방식들의 장점으로는 효율적이고 장애에 대한 관용성, 컨텐츠 소스의 locating 용이, self-organization, self-configuration, self-healing, 높은 확장성, 서비스 신뢰성, 그리고 유지 관리에 대한 용이성 및 로드 밸런싱 등이 있다. 본 논문에서 제안하는 CDN 구조는, 컨텐츠 소스의 등록, 검색, 선택 동작과정에서 이러한 P2P 알고리듬을 사용하므로, 중앙집중형의 CDN에 비해 위와 같은 장점을 가지게 된다.

또한, 중앙집중형 CDN에서는 서비스 제공자 측의 특정 노드가 가용한 소스 노드들 중에서 하나를 선택해서 사용자의 요청에 응답하는 데에 비해, 본 논문에서 제안하는 분산형 CDN에서는 사용자의 요청이 오버레이 네트워크상에서 P2P 알고리듬에 의해 컨텐츠 소스 노드들로 전달되고, 서비스 정책 등을 만족하는 서비스 가능한 소스 노드들이 각종 파라미터와 함께 응답을 하면, 사용자 노드에서는 이 응답들 중에서 사용자 관점에서 서버를 선택하게 되므로, 실시간 사용자 중심의 서버 선택을 할 수 있다는 장점도 있다.

본 논문에서는 분산 구조의 CDN을 제안하기 위해 DHT를 이용하는 P2P 방식 중에서 Pastry를 본 목적에 맞게 수정하여 적용하였으며, 또한 탐색된 서버들 중에서 적합한 서버를 선택하는 알고리듬을 제안하였다.

본 논문의 구성은 다음과 같다. 2장에서는 CDN 및 서버 선택과 관련된 연구에 대해서 기술하였고, 3장에서는 CDN에 수정된 Pastry를 적용하여 분산 형태로 서버를 선택하는 알고리듬을 제안한다. 4장에서는 제안한 알고

리즘에 대한 시뮬레이션 모델과 결과를 기술하고, 마지막으로 5장에서 결론을 맺는다.

II. 관련연구

CDN에 대한 연구의 예로서는 각각의 작은 단위의 CDN들을 하나의 큰 단위 CDN으로 취합하여 구성하는 CCDN(Collaborative Content Delivering Network)[3], 이종 환경의 콘텐츠들을 적응적으로 전달하는 ACDN (Adaptive Content Delivering Network)[4], CDN의 구축 비용을 감소시키기 위한 DCDN(Distributed Content Delivering Network)[5] 등이 있다. DCDN은 넓게 분포된 CDN을 광대역 인터넷에 연결된 전체 인터넷 유저들에게 제공하기 위한 기술이다. 이 모델은 새로운 인프라의 설치에 많은 비용을 필요로 하지 않는다. DCDN 모델은 경제적 비용을 줄이는 효과를 가지며, 적절한 요금체계를 통해 DCDN 대리시스템(surrogate) 사이의 이익 효과를 기대할 수 있다. 또한, 기존의 CDN에서 지원하지 못하는 자원 공유 기능을 DCND 대리시스템의 부가기능으로 지원하게 된다. 지역시스템에서 대부분의 콘텐츠 분배가 이루어지고, 클라이언트는 대리시스템을 통해 원하는 콘텐츠를 얻을 수 있으므로 응답시간에 대한 복구 효율성증가와 트래픽의 감소효과를 기대할 수 있는 구조를 갖는다.

ITU-T나 IETF와 같은 표준화 단체에서도 CDN에 대해 표준화 활동이 진행되고 있다. ITU-T 문서에 제시된 구조에서는 CDN 서버들을 클러스터화 하고 이러한 클러스터를 관리하는 클러스터 제어 서버를 둔다. 또한, 이러한 클러스터 제어 서버들을 전체적으로 관리하는 컨텐츠 전달 및 분산 제어 서버를 배치한다. 즉, CDN 서버들을 클러스터화 한 뒤 이를 계층적이며 중앙집중 방식으로 관리하는 서버들을 배치하는 방식이다[6]. 또한, 사용자간의 P2P방식의 콘텐츠 전달을 중앙에서 관리는 방식인 Hybrid CDN-P2P에 대한 구조와 시나리오에 대해서도 표준화 작업을 진행하고 있다[7].

IETF에서는 기존 CDN방식과는 다르게 사용자에게 콘텐츠를 제공할 때 서로 다른 사업자의 CDN사이에서도 콘텐츠 공유가 일어 날 수 있게 하는 즉, 다수의 CDN 간의 상호 연결을 가능하게 하기 위한 CDN 사이의 프로토콜 및 인터페이스 표준화 작업(CDN Interconnection)을 진행하고 있다[8].

그림 1은 ITU-T 권고안 Y.2010 에 기술된 CDF(content delivery function)의 기능적 구조도를 보인 것이다. CDF는 이 구조를 통해 상위의 AF(application function)과 연동하며, 컨텐츠의 배치 및 복제배치, 전달과 삭제(ingestion, dispatch, age, delivery)의 기능을 담당하게 된다.

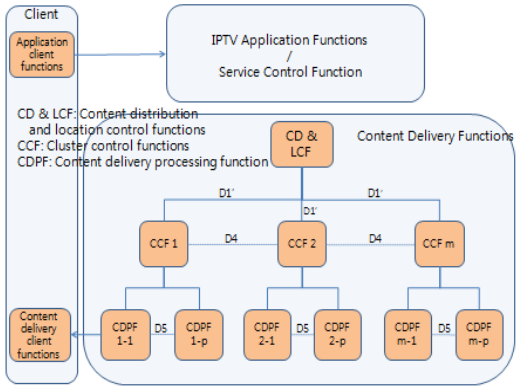

그림 1. 콘텐츠 전달방식의 기능적 구조도
Fig. 1. Functional architecture of contents delivery

본 논문의 주제인 컨텐츠의 선택 및 전달과 관련한 동작을 살펴보면, CD&LCF(Content distribution and location control function)은 컨텐츠의 배치, 선택 IPTV 단말 기능으로의 전달을 최적화하기 위한 컨텐츠 전달 및 저장 기능을 제어한다. CDPF(Content delivery processing function)은 실제 컨텐츠를 단말에 전달하는 서비스를 담당하며, 그림에서와 같이 복수의 CDPF들을 하나의 클러스터로 그룹화하고, 각각의 클러스터에는 하나씩의 CCF(cluster control functions)가 있어서 클러스터 내의 CDPF들을 제어하게 된다.

컨텐츠의 서비스는 단말 사용자의 ACF(application client function)가 서비스 망측의 AF(application function) 기능들과 연동하여 컨텐츠 서비스를 요청하면, 이러한 요청은 CD&LCF를 거쳐 선택된 클러스터의 CCF에 전달되고 CCF는 자기 클러스터 내의 CDPF중 하나를 선택하여 세션 요청을 보내게 된다. 이 때 서비스를 위해 선택된 CDPF가 해당 컨텐츠를 가지고 있지 않은 경우에는 CCF는 다른 클러스터의 CCF에게 컨텐츠 다운로드 요청을 보낸다. 요청 받은 CCF는 자기 클러스트 내의 CDPF를 선택하여 다운로드 요청을 전달하며 이

CDPF가 요청한 클러스터의 CDPF에 컨텐츠를 다운로드 해 준다. 사용자에게 컨텐츠 요청을 받았던 CDPF에 다운로드가 되면, 이 CDPF는 사용자에게 컨텐츠를 서비스하게 된다.

이와 같이, ITU-T 권고안에서의 컨텐츠 전달 기능의 구조는 컨텐츠의 전달을 담당하는 CDPF들을 클러스터화 하고, CCF와 CD&LCF를 이용한 계위구조로서 중앙 집중 형태의 구조를 보여 주고 있다.

서버 선택에 관한 알고리듬은 크게 2가지 카테고리로 구성된다. 첫 번째 카테고리는 네트워크 거리를 이용한 알고리듬[9, 10, 11, 12] 으로서 클라이언트와 서버 사이의 거리 측정을 위해 프로빙 메시지를 이용하는 방법과 IDMaps를 이용한 네트워크 측정 방법, 그리고 클라이언트와 서버 사이의 latency 및 가용 대역폭 측정을 위하여 특정한 툴을 사용하는 방법 등이 있다.

두 번째 카테고리는 서버의 부하와 네트워크 지연에 기반을 둔 알고리듬[13, 14, 15]이다. 이러한 방식에는 HTTP 헤드 메시지를 서버에게 주기적으로 송신하여 RTT 및 홉 수를 측정하는 방법과, Dynamic probe approach를 통한 대역폭 및 latency를 측정하는 방법이 있으며 이를 통해 동적으로 서버를 선택하여 load balancing을 달성하게 된다.

논문[16]에서는 Network의 RTT를 예측하여 이를 토대로 Network Distance를 계산하고 이러한 locality를 기반으로 server set을 구성한다. Locality를 계산하는 방법은 우선 Host에서 각 landmark로 probing packet을 전송하여 RTT를 측정하고 각 Host들은 content를 가지고 있는 server에 location vector를 부여한다. 같은 Content를 가진 여러 개의 Host를 발견하면 location vector를 비교하여 거리적으로 가까운 target server를 선택한다. [17]에서는 전송시간을 예측하여 각 server마다 rank 값을 부여하고 sorting 하여 best server를 선출하는 방식을 제안하였다.

이러한 연구들에서는 그림 1에서와 같은 중앙 집중형 태의 컨텐츠 전달망 구조를 가정하고 있으며, 본 논문에서는, 이와는 달리 DHT(distributed hash table)을 이용한 분산 컨텐츠 전달망(DCD, distributed content delivery)의 구조와 서버 선택 방안을 제안한다.

Ⅲ. 분산 컨텐츠 전달망 및 서버 선택 방안

1. 분산 컨텐츠 전달망 구조

그림 2에 DHT(distributed hash table)을 이용한 분산 컨텐츠 전달망의 구조를 보였다. 그림 1의 중앙 집중 형 구조와는 달리 컨텐츠 소스 노드들은 각 컨텐츠의 키를 DHT를 이용하여 키 루트에 등록한다. 그리고 사용자의 컨텐츠 서비스 요청은 중앙 집중된 하나 혹은 소수의 처리 노드에 집중되는 것이 아니라, 키 루트 도메인에서 DHT를 통해 해당 컨텐츠의 키 루트로 보내진다. 키 루트에는 해당 컨텐츠를 가진 소스 노드들이 등록되어 있으므로, 키 루트는 이 소스 노드들에 프로빙 요청을 보낸다. 이 요청을 받은 각 소스 노드는 서비스 요청을 한 사용자 요청 노드에 프로빙 응답을 보내고, 요청 노드는 이들 응답 중에서 응답 메시지 내의 파라미터들을 보고 선택 기준에 의거해서 서비스를 하게 될 서버 노드를 선택하게 된다.

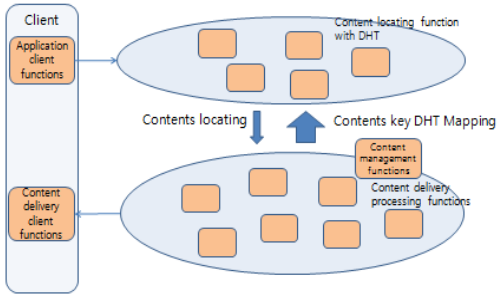

그림 2. 제안하는 DHT를 이용한 분산 구조
Fig. 2. Proposed distributed architecture using DHT

중앙 집중 형과 분산 형 구조는 서로 장단점이 있지만, 분산 형 구조의 장점은 확장성, 강건성, 노드의 장애 시 관용성, 효율성, 서비스 요청 처리 부하의 분산, 트래픽의 분산 등이 있다.

2. Pastry

Pastry[21]는 확장성이 있고 장애에 강하며 셀프-오거나이징(Self-Organizing)의 P2P 기법으로서, Chord와 유사한 주소공간을 가지지만 주소 공간의 크기가 128bit로 고정되어 있다. 각각의 노드들은 원형 주소공간에서 유일하고 균등하며 랜덤하게 할당된 노드 id를 가진다. 또한 Pastry의 노드 id와 키는 2b(b는 일반적으로 4를 갖는

설정 파라미터)진법이라는 임의의 진법을 사용한다. 각각의 노드들은 Leaf set, Routing table, Neighborhood set의 3개의 테이블을 가지고 있다.

각각의 테이블들은 라우팅을 위한 정보들로서 Leaf set은 주소 공간 내에서 현재 노드에 가장 근접한 L개의 노드들의 정보들을 유지한다. 자신의 id를 기준으로 L/2개의 자신의 id보다 큰 노드 id들과 L/2개의 자신의 id 보다 작은 id들의 정보를 가지고 있다. Routing table은 행렬을 가지며 이 행렬에 노드들의 정보를 가지고 있다. 각 행은 자신의 id와 다른 노드들의 id 사이의 공통 프리픽스(prefix) 길이가 짧은 노드들을 먼저 배치하고 길이가 긴 노드들을 행의 아래쪽에 배치한다. 즉 1행에는 공통 프리픽스가 없는 노드들의 정보를 가지고 있고 2행에는 공통 프리픽스가 1자리인 노드의 정보들을, 3행에는 공통 프리픽스가 2 자리인 노드의 정보들을 가지는 방식으로 테이블이 구성된다.

마지막 테이블인 Neighborhood set은 Leaf set과 마찬가지로 L개의 노드들의 정보를 유지하는데 프락시머티 매트릭(proximity metric)에 따라 현재 노드와 네트워크상으로 가장 근접한 노드들의 정보를 포함하고 있다.

Pastry에서의 라우팅 과정을 살펴보면 다음과 같다. 우선 위의 3가지 테이블 중에서 가장 먼저 Leaf set를 찾아서 주소 공간 내에서 가장 근접한 노드를 검색하게 된다. 이 Leaf set에서 원하는 주소를 찾지 못한다면 Routing table의 정보를 찾아보게 되고 가장 긴 공통 프리픽스를 가지는 노드로 메세지를 전송하여 라우팅을 실행하게 된다. Routing table에서도 노드를 찾을 수 없다면, 현재 노드가 가지고 있는 정보 중 목표 노드의 주소에 가장 근접한 주소를 가진 노드에 메시지를 포워딩하여 라우팅하게 된다.

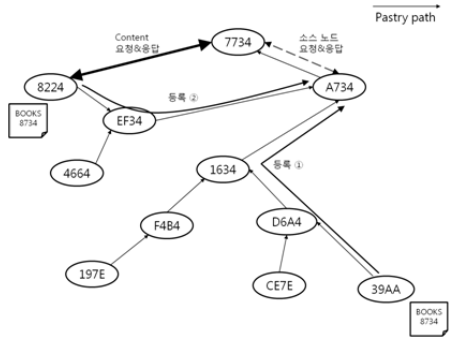

그림 3. Pastry 동작과정
Fig. 3. Operation Procedure of pasty

그림 3은 pastry의 동작 과정을 나타내고 있다. 컨텐츠를 가지고 있는 소스 노드는 자신의 컨텐츠의 제목 등의 정보를 해쉬한 값을 아이디로 가지는 오버레이 노드(키 루트)에 자신의 위치 정보를 저장한다. 이 때 중복되는 컨텐츠를 등록하게 되면 마지막으로 등록하는 소스 노드의 위치 정보로 갱신된다.

따라서 클라이언트가 해당 컨텐츠의 소스 노드를 요청하게 되면 마지막으로 등록한 소스노드의 위치정보를 알려주게 되고 클라이언트는 알게 된 소스노드에게 콘텐츠를 요청하게 된다. pastry 관련의 논문에는 복수개의 소스노드를 등록할 수 있는 여지에 대해 언급한 바는 있으나 구체적인 동작에 대한 기술은 없다.

3. 제안된 서버선택 방안

본 논문에서는 DHT를 이용한 분산 CDN에서 서버를 선택하는 방안을 제안하기 위해 상기의 pastry를 변형한 알고리듬을 활용한다. 우선, 복수의 소스 노드를 키 루트 노드에 등록하고, 복수의 소스 노드 중에서 하나의 소스 노드를 최종 서비스할 서버로 선택하기 위해 키 루트는 복수의 소스 노드들에게 프로빙 요청(probing request)을 보낸다. 이 프로빙 요청을 받은 각 소스 노드는 검색 요청 노드에게 프로빙 응답을 보내게 되고, 검색 요청 노드는 이들 프로빙 응답들을 수신하여 그 중에서 적합한 소스 노드를 서버로 선택하게 된다.

각 소스가 요청 노드에 프로빙 응답을 보낼 때에는 그 메시지 내에 서버 선택의 기준으로 사용될 수 있는 각종 파라미터들을 기재한다. 여기에 사용될 수 있는 파라미터들로는 각 소스에서 요청 노드에 이르는 언더레이 네트워크(underlay network)의 홉(hop)수나 전파지연(propagation delay), 혹은 각 소스의 처리(processing) 부하, 링크의 트래픽 부하 등 다양한 기준들이 사용될 수 있으며, 요청 노드는 이들 파라미터 중에서 하나 혹은 여러 개를 사용한 선택 기준을 사용할 수 있다. 본 논문에서는 언더레이 네트워크의 홉수 및 전달지연를 이용하여 소스 노드 선택 방안을 기술한다.

그림 4는 수정된 pastry의 동작 과정이다. 컨텐츠를 가지고 있는 소스 노드는 자신의 컨텐츠 제목 등의 정보를 해쉬한 값을 아이디로 가지는 오버레이 노드에 자신의 위치 정보를 저장한다.

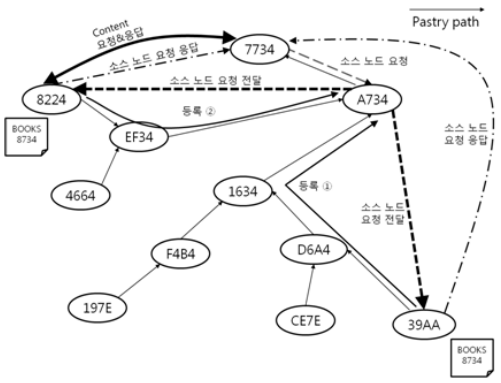

그림 4. 수정된 pastry 동작 과정
Fig. 4. Modified operation Procedure of pasty

이 때 중복되는 컨텐츠를 등록하게 되면 기존의 pastry와는 달리 모든 소스 노드의 위치 정보를 등록한다. 이 후 클라이언트가 해당 컨텐츠의 소스 노드를 요청하게 되면 등록되어 있는 모든 소스노드에게 클라이언트의 요청을 전달하고 클라이언트의 요청을 받은 소스 노드들은 타임스탬프 등 파라미터들을 포함하여 클라이언트에게 응답한다. 클라이언트는 소스 노드들로부터의 응답을 바탕으로 자신이 원하는 QoS를 만족하는 소스 노드를 선택할 수 있게 된다.

기존의 중앙집중 방식의 서버 선택에서는 집중된 어떤 노드에서 서버를 선택해서 요청 노드에게 알려 주는 형태이지만, 제안된 방식에서는 서버의 검색이 분산 해시 테이블을 이용하여 분산형태로 진행되며, 또한 각종 파라미터를 기반으로 검색 요청 노드의 입장에서 서버를 선택할 수 있게 된다.

표 1. 서버 선택 알고리듬
Table 1. Algorithm of server selection

1. Client, C sends ObjRoot for Object through p2p
2. ObjRoot sends ObjQuery to ObjSource(s)
3. ObjSources, OS_i that receive ObjQuery
   if(utilization $U_i < U_{max}$)
   Send ObjReply(U_i) to C with traveling limit
4. C waits for time, t_w for ObjReplys
5. C collects ObjReplys and estimates distance, d to each ObjSource
   if C finds OS(s) with d(C, OS_i) < d_{max}
   Choose OS_m with min(d)
   C sends ObjReq to OS_m
6. OS_m receives ObjReq
7. OS_m sends CertfQuery to SDN node
8. SDN node that receives CertfQuery
   if OS_m CertfQuery complies SDN policy
   sends service permission, SvcConfirm to OS_m
   else
   sends service denial, SvcDeny to OS_m
9. OS that receive SvcConfirm starts service process to C

표 2. 알고리듬 notation
Table 2. Notation of algorithm

Notation	설명
OR	오브젝트를 가지고 있는 서버에 대한 요청
OS	오브젝트 소스로부터의 응답
OSm	dmax 보다 적은 delay를 가지면서 클라이언트로부터 물리적으로 가장 근접한 OS
ObjQuery	오브젝트를 가지고 있는 서버에 대한 요청
ObjReply	OS가 서비스 제공
ObjReq	실제 오브젝트에 대한 요청
CeryQuery	OS가 서비스 제공에 대한 인증을 받기 위해 SDN노드에게 보내는 쿼리
SvcConfirm	SDN노드가 OS의 서비스 제공을 허락하는 확인 메시지
SvcDeny	SDN노드가 OS의 서비스 제공을 거부하는 메시지
Ui	노드의 현재 이용률
Tw	ObjReply를 기다리는 대기 시간

표 1은 수정된 pastry가 적용된 CDN에서 서버 선택 알고리즘 절차에 대해 나타내고 있고, 표 2는 알고리즘의 기호 정보를 나타내고 있다. 클라이언트는 망의 액세스 노드를 통해 오브젝트 쿼리를 보낸다. 이 때 수정된 P2P 알고리즘을 기반으로 오브젝트의 루트 노드에게 클라이언트의 쿼리를 보내게 되는데 본 논문에서는 상기의 수정된 Pastry를 사용한다.

쿼리를 받은 오브젝트의 루트 노드는 자신의 오브젝트 소스 리스트 중 요청하는 오브젝트를 보유하고 있는 모든 소스들에게 오브젝트 쿼리를 전달한다.

오브젝트 쿼리를 전달받은 오브젝트 소스들 중 자신의 현재 이용 율이 한계를 넘지 않거나, 또는 요청되는 오브젝트의 오리지널 소스인 경우 쿼리에 대한 응답을 보내게 된다. 이 때 응답메시지에는 자신의 현재 이용 율과 이동 한계에 대한 정보를 포함한다.

클라이언트는 tw 의 시간동안 소스로부터의 응답을 기다린 후 tw 시간 내에 응답을 보내온 소스들과의 distance를 ping 메시지를 통해 측정한다. 이 결과를 토대로 클라이언트는 자신이 요구하는 delay QoS를 만족하는 즉, dmax보다 적은 distance를 가지는 소스들을 찾

아 오름 차 순으로 정렬한 list를 작성한다. List의 가장 상위에 있는 소스가 가장 낮은 delay를 가지는 소스가 되므로 클라이언트는 이 소스에게 오브젝트 요청 메시지를 보내게 된다. 오브젝트 요청을 받은 소스는 소스들을 관리하는 서버(SDN 노드)에게 서비스 제공에 대한 인증 쿼리를 보내게 된다. 쿼리를 받은 관리 서버는 미리 정의된 정책(resource사용, load 한계 등)에 합당할 경우 쿼리를 보낸 소스에게 서비스를 허락하는 확인응답을 보내게 된다. 만약 정책에 합당하지 않는 경우 쿼리를 보낸 소스에게는 서비스 거부 메시지를 보낸다. 서비스 확인 응답을 받은 소스는 클라이언트로의 서비스 제공을 시작한다.

본 논문에서 제안하는 서버 선택 알고리듬은 P2P를 이용하여 오브젝트를 보유하고 있는 서버들을 찾아낸 후 이들 중 사용자의 QoS를 만족하면서 망 사업자나 서비스 제공자의 정책에 합당한 서버를 선택하여 서비스를 제공하게 함으로써 사용자의 QoS 뿐만 아니라, 망 사업자 및 서비스 제공자의 입장도 고려한 알고리듬이다. 즉, 분산형태의 구조를 CDN에 적용하였으며, 또한, 중앙 집중 방식도 고려하여 관리 서버를 배치함 으로써 Managed Distributed의 CDN을 제안하였다. 이를 통해 사용자 측면에서는 사용자와 가장 근접한 서버를 선택함으로써 서비스 제공시의 딜레이를 줄일 수 있고, 네트워크 측면에서는 서비스를 제공하는 서버들의 부하를 균등하게 분산시켜 네트워크 load-balancing를 할 수 있다는 장점이 있다.

Ⅳ. 시뮬레이션 성능

1. 시뮬레이션 환경

서버 선택 알고리듬의 성능 분석을 위해 OMNeT++[22]을 이용하여 시뮬레이션 하였다. Underlay 토폴로지 구성을 위해 ReaSE프레임워크[23]를 사용하였고, 네트워크 계층 프로토콜 사용을 위해 Inet 프레임워크[24]를 사용하였다. 또한, P2P 알고리즘 적용을 위해 Oversim 프레임워크[25]를 사용하였다. 시뮬레이션 구성도는 그림 5와 같다.

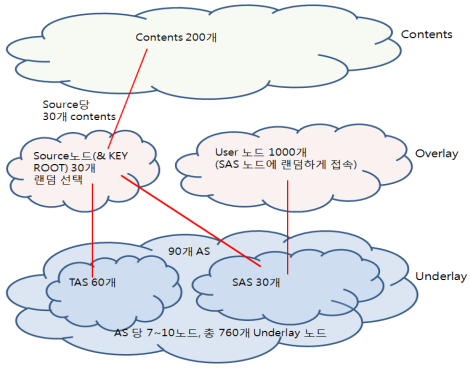

그림 5. 시뮬레이션 구성도
Fig. 5. Architecture of Simulation

2. 시뮬레이션 토폴로지 설정

제안된 분산 서버 선택 구조 및 방안을 시뮬레이션을 하기 위해 다음과 같이 망의 토폴로지를 구성하였다. AS의 개수는 총 90개로 각 AS에는 7~10개의 edge router를 배치하였다. 가장 끝단에 배치되는 SAS(Stub Autonomous System)에는 사용자 역할을 하는 requester 오버레이 노드 1000개를 랜덤하게 배치하였다. 또한, 서버 및 key root의 역할을 담당하는 source 오버레이 노드 30개는 SAS와 TAS(Transit Autonomous System)에 랜덤하게 배치하였다.

컨텐츠는 서로 다른 200개가 존재한다고 가정하였고, 각각의 source는 200개의 컨텐츠 중 30개의 서로 다른 컨텐츠를 put 한다고 가정하였다.

시뮬레이션 과정은 1030개(source = 30개, requester = 1000개)의 조인을 완료하고 30개의 source는 각자 30개의 컨텐츠를 모두 put한다. 이 후 1000개의 requester가 get을 통해 컨텐츠를 요청한다. 시뮬레이션 전체 토폴로지는 그림 6과 같다.

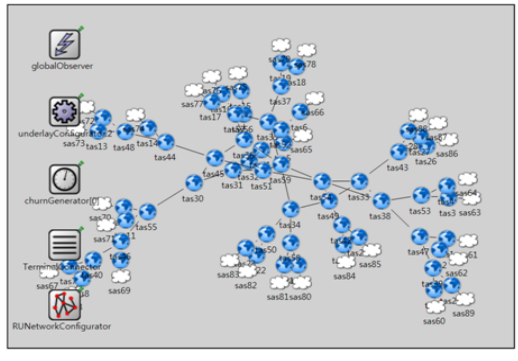

그림 6. 시뮬레이션 토폴로지
Fig. 6. Topology of Simulation

각각의 AS는 7~10개의 edge router을 가지고 있고 새롭게 조인하는 오버레이 노드들은 각 edge router에 연결된다. 각 AS 내부는 그림 7과 같이 구성된다.

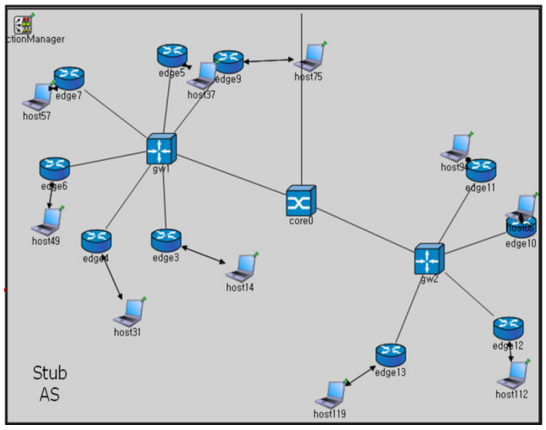

그림 7. AS 내부 토폴로지
Fig. 7. Inner topology of AS

표 3. 토폴로지 발생 파라미터
Table 3. Parameter of Topology Generation

Parameter		Value
Underlay AS num		90
Underlay Node (edge router) num		760
Link setting	Host - Edge	BW : 1Mbit/s
	Edge - Host	BW : 1Mbit/s
	Server - Edge	BW : 10Mbit/s
	Edge - Gateway	BW : 155Mbit/s
	Gateway - Core	BW : 622Mbit/s
	Core - Core	BW : 1000Mbit/s
	Stub - Stub	BW : 1000Mbit/s
	Transit - Stub	BW : 5000Mbit/s
	Transit-Transit	BW : 100000Mbit/s

표 3은 ReaSE 프레임워크를 사용하여 구성한 Underlay Network 토폴로지의 파라미터들을 나타낸 것이다. 전체 AS 개수 및 각 link의 대역폭에 대해 나타내었다.

표 4. 시뮬레이션 동작을 위한 파라미터
Table 4. Parameter of Operating simulation

Parameter		Value
Overlay Node num	Source& root	30
	requester	1000
Overlay node join interval		0.1s
Total content num		1000
Put content num per overlay node(source)		30
Queue length		100000
Tw		2.3s
Message interval	Put	10~20s
	Get	55~65s
Total simulation time		5000s

표 4는 실제 시뮬레이션을 위한 파라미터를 나타내고 있다. 조인하는 오버레이 노드 수 및 조인 간격, 총 컨텐츠 수, 소스당 put 하는 컨텐츠 수, 큐 길이, 응답 대기 시간 등이 있다.

3. 시뮬레이션 결과

시뮬레이션에서는 5가지의 다른 방안을 비교해 보았다. 첫 번째 방안(SSS, single souce selection)에서는 키 루트에 최종 등록된 하나의 소스만 등록하게 하고, 검색 요청이 오면 키 루트는 이 등록된 소스 하나에만 요청을 보내고, 이 소스가 요청 노드에 서비스를 한다. 두 번째 방안(FAS, first arrival selection)에서는 키 루트가 복수의 소스들에게 프로빙 요청을 보내고, 요청 노드는 소스들로부터 도착하는 프로빙 응답 중에서 가장 먼저 도착하는 소스를 선택한다. 세 번째 방안(WBS, wait for better selection)에서는 일정시간을 대기하여 그 대기시간 내에 도착하는 프로빙 응답 중에서 소스에서 요청 노드까지의 언더레이 네트워크 홉 수 가 가장 적은 것을 선택한다. 네 번째 방안(COP, caching on the path)에서는 소스로부터 키 루트로 put 할 시 오버레이 3 홉 까지 소스 노드의 정보를 저장한다. 소스 선택은 WBS와 동일하다. 다섯 번째 방안(CFA, caching and first arrival selection)은 COP 방법 중 요청 노드가 가장 먼저 도착하는 소스 노드를 선택 한다.

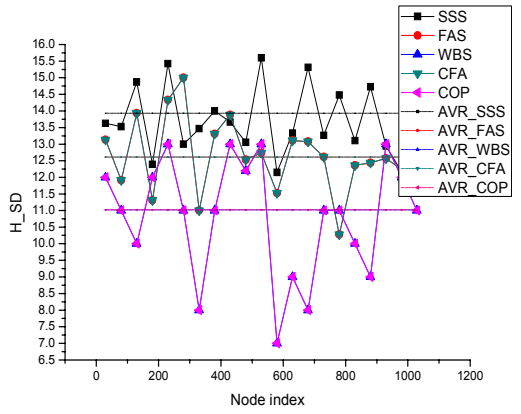

그림 8. Source에서 Destination으로 갈 때의 홉수
Fig. 8. Number of hops from Source to Destination

그림 8에는 5가지 방식에서의 선택된 소스와 요청 노드 사이의 언더레이 네트워크 홉 수를 비교하였다. 가로축에는 요청 노드들의 번호(ID)를 나타내었는데, 편의상 전체 요청 노드에 대한 홉 수의 평균과, 21개의 노드만 샘플하여 그 결과를 보여 주고 있다. 그림에 보인 것과 같이 WBS와 COP 방식이 가장 좋은 결과를 보여 줌을 알 수 있다. Caching이 적용되는 COP와 CFA가 각각 WBS, FAS와 같은 이유는 Caching을 할 시 전체 경로의 홉 수는 줄어들지만 소스 노드로부터 사용자 요청 노드로의 홉 수와 전파 지연은 영향을 받지 않는다. WBS, COP 방식에서는 소스들의 프로빙 응답을 기다리는 대기시간이 있는데, 이 대기시간이 길면 더 많은 소스의 프로빙 응답을 받아서 더 나은 선택을 할 수 있겠지만, 서비스 지연(latency)이 길어지므로 무작정 길게 할 수는 없다.

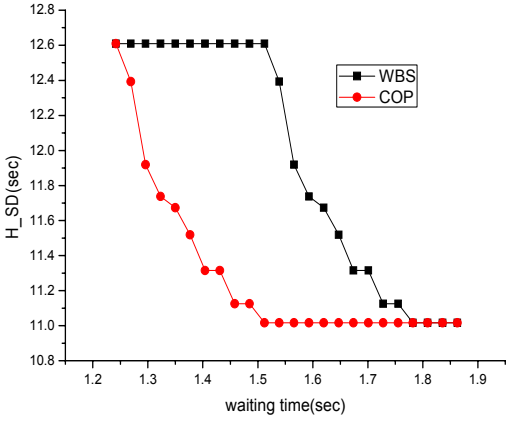

그림 9. Wating time에 따른 서버-유저 홉수 영향(MS)
Fig. 9. According to Wating time, Effect of number of hops between Server and User (MS)

그림 9 에서는, 이 대기시간의 증가에 따른 선택된 소스 노드와 요청노드 간의 언더레이 노드 홉 수를 나타내었다. 이 그림에서 대기시간이 어느 정도 이상 길어지면 더 이상 홉 수 거리의 개선은 포화되는 것을 볼 수 있다.

앞에서 설명한 대로 COP의 경우 전체 경로의 홉 수는 줄어들지만 소스 노드에서 사용자 요청 노드까지의 홉 수 는 영향을 받지 않는다. 그러므로 대기 시간은 줄어들지만 홉 수는 WBS와 동일하다. 그림 9의 결과로 WBS와 COP 방안에서는 적절한 대기시간을 선택하는 것이 필요하다는 것을 알 수 있으며, 본 논문에 보여 진 그래프의 WBS 방식에서는 대기시간을 1.782sec , COP 방식에서는 1.512sec로 설정하였다.

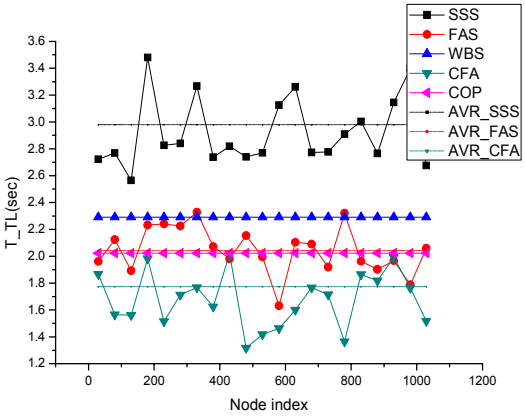

그림 10. Request에서 service까지의 총 서비스 latency
Fig. 10. Total Service Latency from request to service

그림 10에는 5가지 방안의 총 서비스 지연시간을 보였다. 그림 8과 그림 10의 결과를 종합할 때, COP 방안은 사용자의 총 서비스 지연 시간에 큰 영향을 주기 않으면서, 본 논문에서 사용자 입장 QoS로 채택한 파라미터인 소스와 요청 노드간의 언더레이 홉 수에 개선을 가져올 수 있음을 알 수 있다.

V. 결론

본 논문에서는 기존의 중앙집중 형태의 CDN 방식과는 다른 분산형 CDN 구조 및 서버 선택방안을 제안하였다. 중앙집중 형 CDN에서는 사용자의 요청에 대해서 서비스 제공자 측에서 망 내에 가용한 서버 중에 하나를 선택하여 사용자에게 알려 주는 데에 비해서, 제안한 분산

형 방식에서는 사용자의 요청이 오베레이 망에서 DHT를 이용한 P2P 알고리듬에 의해 분산방식으로 소스 노드들에 전달되고, 소스 노드들이 응답을 하면, 사용자 노드는 실시간 사용자 중심의 각종 기준에 의해 서버를 선택할 수 있게 된다. 또한 분산형 구조가 가지는 부하의 분산, 트래픽의 분산, 자가 구성 및 회복에 의한 장애에 대한 관용성 등 다양한 장점도 가진다.

본 논문에서는 이러한 분산형 구조 CDN에서의 컨텐츠 등록, 검색 및 선택 방식을 위해 DHT(Distributed Hash Table)를 이용한 수정된 Pastry를 제안하고 적용하였으며, 이를 이용한 사용자의 QoS를 만족하는 최적의 서버를 선택하는 서버 선택 알고리듬을 제안하였다. 성능 분석을 위하여 OMNeT++를 이용한 시뮬레이션을 통해 몇 가지의 변형 시나리오의 성능을 비교하였다.

제안한 알고리듬은 다수의 소스 서버가 존재하는 경우 전달지연, 홉수, 서버의 부하 등 QoS 및 서비스 정책에 부합되는 다양한 선택 기준을 적용하여 서버를 선택할 수 있으며, 본 시뮬레이션에서는 사용자와 가장 작은 전달지연이 측정되는 컨텐츠 소스 서버를 선택하였다.

본 논문에서는 제안한 구조와 수정된 Pastry 알고리듬에 대해, 단일 소스 노드의 경우와, 그리고, 복수의 소스 노드 등의 경우에 대해서 시뮬레이션 하였다. 또한, 복수 소스 노드의 경우에도, 가장 먼저 응답이 오는 서버를 선택하는 방안, 일정 시간을 기다려 가장 짧은 소스-사용자 간 전달지연을 갖는 소스를 선택하는 방안, 컨텐츠의 등록 시에 소스와의 일정 홉수 내의 노드가 캐싱을 하는 방안, 또한, 캐싱 방안에 대해서도 두 가지 변형에 대해 시뮬레이션을 하고 선택된 소스들의 전달지연, 홉수, 그리고, 요청에서 서비스 시작까지의 총 지연시간 등을 비교하였다.

시뮬레이션 결과, 컨텐츠 소스 주변의 노드가 소스 등록 시에 캐싱을 하고, 사용자가 일정시간을 기다려서 회신된 응답 중에서 선택하는 방안이 전달지연 측면에서 가장 좋은 성능을 보임을 확인할 수 있었다.

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

[3] G. Pierre and M. van Steen. “Globule: A collaborative content delivery network”, IEEE Communications Magazine, pages 127 - 133, August 2006.

[4] J. Coppens et al., “Design and performance of a self organizing adaptive content distribution network”, IEEE/IFIP Network Operations Management Symposium 2006, Vancouver, Canada, April 2006.

[5] H. J. Mulerikkal, et al., ″An Architecture for Distributed Content Delivery Network″, IEEE 2007.

[6] ITU-T Rec. Y.2019 “Content delivery functional architecture in NGN” (2010.09)

[7] ITU-T Rec. Y.1902 “Framework for multicast based IPTV content delivery” (2011.04)

[8] IETF 80, cdni BoF (2011.03)

[9] A. Guyton and M. Schwartz, “Locating nearby copies of replicated internet servers,” in Proc. ACM SIGCOMM, August 1995, pp. 288 - 298.

[10] R. L. Carter and M. E. Crovella, “Server selection using dynamic path characterization in wide-area networks,” in Proc. IEEE INFOCOM, April 1997, pp. 1014 - 1021.

[11] S. Jamin, C. Jin, Y. Jin, D. Raz, Y. Shavitt, and L. Zhang, “On the placement of internet instrumentation,” in Proc. IEEE INFOCOM, March 2000, pp. 295 - 304.

[12] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker, “Topologicallyaware overlay construction and server selection,” in Proc. IEEE INFOCOM, June 2002, pp. 1190 - 1199.

[13] M. Sayal, Y. Breitbart, P. Scheuermann, and R. Vingralek, “Selection algorithms for replicated web servers,” ACM SIGMETRICS Performance Evaluation Review, vol. 26, no. 3, pp. 44 - 50, 1998.

[14] S. G. Dykes, K. A. Robbins, and C. L. Jeffery, “Empirical evaluation of client-side server selection algorithms,” in Proc. IEEE INFOCOM, March 2000, pp. 1361 - 1370.

[15] K. M. Hanna, N. Natarajan, and B. N. Levine, “Evaluation of a novel two-step server selection metric,” in Proc. IEEE International Conference on Network Protocols (ICNP), November 2001, pp. 290 - 300.

[16] S. Naoto, K, Wataru, W Hiroshi, “A Novel Content Distribution Architecture Utilizing Network Distance Prediction”, Waseda University, Tokyo, Japan, 2003.

[17] M. Malli, C. Barakat, W. Dabbous, “An Efficient Approach for Content Delivery in Overlay Networks”, CCNC’05, 2005.

[18] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content addressable network. In Proceedings of ACM SIGCOMM 2001, 2001.

[19] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In Proceedings of the 2001 Conference on applications, technologies, architectures, and protocols for computer communications. ACM Press New York, NY, USA, 2001.

[20] B. Godfrey, K. Lakshminarayanan, S. Surana, R. Karp, and I. Stoica. Load balancing in dynamic structured P2P systems. In Proc. IEEE INFOCOM, Hong Kong, 2004.

[21] Rowstron, P. Druschel, “Pastry: Scalable, Distributed Object Location and Routing for Large-scale Peer-to-Peer Systems”, in IFIP/ACM Int’l Conf. on Distr. Systems Platforms(Middleware) 2001, pp.329-350, 2001.

[22] A. Varga. Omnet++ community site. [Online]. Avalaible:http://www.omnetpp.org/

[23] Thomas Gamer, Michael Scharf, “Realistic Simulation Environments for IP-based Networks” , OMNeT++ 2008, March 3, 2008, Marseille, France

[24] INET Framework. http://www.omnetpp.org/pmwik/index.php?n=Main.INETFramework, Sept. 2007.

[25] I. Baumgart, B. Heep, and S. Krause, “OverSim: A Flexible Overlay Network Simulation Framework”, Proceedings of 10th IEEE Global Internet Symposium (GI ′07) in conjunction with IEEE INFOCOM 2007, Anchorage, AK, USA, May 2007.

저자 소개

정 종 해(비회원)

- 2002-2009 : 한국항공대학교 통신공학과(공학사)
- 2009-2011 : 한국항공대학교 통신공학과(공학석사)

오 건 영(비회원)

- 2003-2010 : 한국항공대학교 통신공학과(공학사)

이 남 경(비회원)

- 1996 : 한국항공대학교 학사(공학사)
- 1998 : 한국항공대학교 컴퓨터공학과 석사(공학석사)
- 2005 : 한국항공대학교 컴퓨터공학과 박사(공학박사)
- 2001년 ~ 현재 : 한국전자통신연구원 선임연구원

윤 장 우(비회원)

- 1990 : 서강대학교 학사(공학사)
- 1992 : 포항공과대학교 대학원 석사(공학석사)
- 2005 : University of Florida 컴퓨터공학 박사(공학박사)
- 1992년 ~ 현재 : 한국전자통신연구원 책임연구원
- 2008년 ~ 현재 : UST 광대역네트워크공학 겸임교수

이 현 우(비회원)

- 1993 : 한국항공대학교 학사(공학사)
- 1995 : 한국항공대학교 정보통신공학과 석사(공학석사)
- 2005 : 한국항공대학교 정보통신공학과 박사(공학박사)
- 1995년~현재 : 한국전자통신연구원 스마트스크린융합연구팀 팀장

이 성 창(정회원)

- 1976-1980 : 경북대학교 전자공학과(공학사)
- 1983-1985 : 한국과학기술원 전기 및 전자공학과(공학석사)
- 1987-1991 : Texas A&M University 전기전자공학과(공학박사)
- 1985-1987 : 한국과학기술원 시스템공학 센터
- 1992-1993 : 한국전자통신연구원
- 1993-현재 : 한국항공대학교 항공전자 및 정보통신공학부 교수

<주관심분야 : 방송통신융합, 스마트네트워크>

류 원(비회원)

- 1983 : 부산대학교 학사
- 1987 : 서울대학교 계산통계학과 석서
- 2000 : 성균관대학교 정보처리학과 박사
- 1989~현재 : 한국전자통신연구원 스마트스크린 융합연구부 부장