통합형 무선 인터넷 프록시 서버 클러스터 구조

곽 후 근[†] · 정 규 식[††]

요 약

본 논문에서는 무선 인터넷 프록시 서버 클러스터를 사용하여 무선 인터넷의 문제와 요구들을 캐싱(Caching), 압축(Distillation) 및 클러스터(Clustering)를 통하여 해결하려고 한다. TranSend는 클러스터링 기반의 무선 인터넷 프록시 서버로 제안된 것이나 시스템적인(Systematic) 방법으로 확장성을 보장하지 못하고 모듈간의 불필요한 통신구조로 인해 복잡하다는 단점을 가진다. 본 연구자들은 기존 연구에서 시스템적인 방법으로 확장성을 보장하는 CD-A라는 구조를 제안하였으나 이 역시 모듈간의 부분적으로 불필요한 통신 구조를 가진다는 단점을 가지고 있다. 이에 본 논문에서는 시스템적인 확장성과 단순한 구조를 가지는 클러스터링 기반의 통합형 무선 인터넷 프록시 서버를 제안한다. 16대의 컴퓨터를 사용하여 실험을 수행하였고 실험 결과 TranSend 시스템과 CD-A 시스템에 비해 각각 196%, 40%의 성능 향상을 보였다.

키워드 : 무선 인터넷, 프록시 서버, 클러스터링, 확장성, 복잡성

A Consolidated Wireless Internet Proxy Server Cluster Architecture

Hukeun Kwak[†] · Kyusik Chung[††]

ABSTRACT

In this paper, wireless internet proxy server clusters are used for the wireless internet because their caching, distillation, and clustering functions are helpful to overcome the limitations and needs of the wireless internet. TranSend was proposed as a clustering based wireless internet proxy server but it has disadvantages; 1) its scalability is difficult to achieve because there is no systematic way to do it and 2) its structure is complex because of the inefficient communication structure among modules. In our former research, we proposed the CD-A structure which can be scalable in a systematic way but it also has disadvantages; its communication structure among modules is partly complex. In this paper, we proposed a consolidated scheme which has a systematic scalability and an efficient communication structure among modules. We performed experiments using 16 PCs and experimental results show 196% and 40% performance improvement of the proposed system compared to the TranSend and the CD-A system, respectively.

Key Words : Wireless Internet, Proxy Server, Clustering, Scalability, Complexity

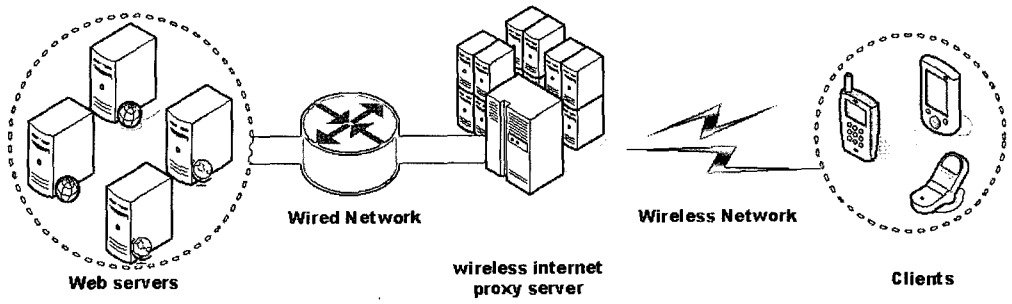
1. 서 론

무선 인터넷에 대한 관심이 증가하는 가운데 핸드폰, PDA 등의 무선 인터넷 단말기의 수요가 늘어나며 보편화 되어가고 있다. 그리고 무선 인터넷 서비스도 기존의 정보검색 위주의 간단한 서비스에서 전자 상거래나 멀티미디어 서비스 등의 복잡한 서비스로 사용자들의 욕구가 상승하고 있다. 그러나 무선 인터넷의 사용이 증가하는 만큼 무선 인터넷의 본질적인 문제 역시 무시할 수 없는 요소로 부각되고 있다. 현재까지 나와 있는 무선 인터넷의 근본적인 문제점은 낮은 대역폭, 빈번하게 연결이 끊김, 단말기 내의 낮은 컴퓨팅 파워 및 작은 화면, 단말기 사용자의 이동성, 네트워크 프로토콜, 보안 등이 있다. 그리고 급속도로 증가하는 수요에 따라 무선 인터넷 서버는 대용량 트래픽을 처리할 수 있는 확장성이 요구되어지고 있다.

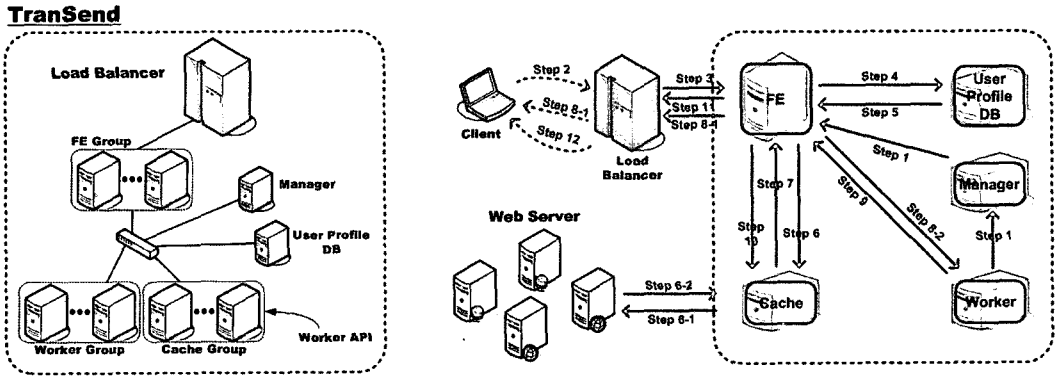
본 논문에서는 이러한 무선 인터넷의 문제점 및 요구들을 캐싱(Caching)[1], 압축(Distillation)[2] 및 클러스터링(Clustering)을 통하여 해결하는 방법으로 클러스터링 기반의 무선 인터넷 프록시 서버를 사용하였다. (그림 1)은 무선 인터넷에 사용되는 무선 인터넷 프록시 서버를 나타내고, 이는 무선 사용자를 유선 인터넷 서버에 연결 시켜주는 역할을 한다.

본 연구자들은 기존 논문[3]에서 기존의 클러스터링 기반의 무선 인터넷 프록시 서버인 TranSend[4]를 개선한 CD-A라는 구조를 제안하였다. 본 논문에서는 TranSend와 CD-A의 구조상의 문제점을 지적하고 이를 해결하는 새로운 구조를 제안한다. 그리고 실험을 통해 이들의 성능을 비

※ 본 연구는 숭실대학교 교내 연구비 지원으로 이루어졌음.
† 준 회 원 : 숭실대학교 전자공학과 대학원 졸업
†† 정 회 원 : 숭실대학교 정보통신전자공학부 교수
논문접수 : 2006년 3월 16일, 심사완료 : 2006년 5월 8일



(그림 1) 무선 인터넷 프록시 서버



(a) 구조 (b) 동작 과정

(그림 2) TranSend 무선 인터넷 프록시 서버

교하는데 초점을 맞춘다.

본 논문의 구조는 다음과 같다. 2장에서는 기존 무선 프록시와 이들이 가지는 문제점을 소개한다. 3장에서는 기존 무선 프록시가 가지는 문제점들을 해결하는 제안된 구조를 설명하고, 4장에서는 실험 및 토론을, 5장에서는 결론 및 향후 연구 방향을 제시한다.

2. 연구 배경

2.1 기존 무선 인터넷 프록시 서버

기존 무선 인터넷 프록시 서버로는 TranSend[4], Mowgli (Mobile Office Workstations using GSM Links)[5], WebExpress[6], Class-based Proxy[7], KWU Proxy[8], Distributed Proxy [9, 10], TranSquid[11], Grid Proxy[12], SHAKE[13] 등이 있고 이들에 대한 자세한 설명 및 비교는 참고 문헌[14]에 기술되어 있다.

기존 무선 인터넷 프록시 서버들은 기본적으로 캐싱(Caching)과 압축(Distillation) 기능을 제공하고 있다. 그러나 확장성 관점과 구조적 관점에서 무선 프록시들 간의 개별적 특성에 따라 차이점을 가지고 있다. 먼저, 확장성 관점에서는 일부[4, 10]는 고려하고 있는 반면에 대부분은 고려하고 있지 않다. 구조적 관점에서도 캐싱과 압축 기능이 대부분 하나의 호스트에 통합(Unity)되어 있지만, 일부[4, 5]는 다른 호스트로 분리(Separation)되어 있다.

2.2 TranSend

(1) 구조

TranSend[4] 구조는 각 모듈로써 Front End(FE, MS : Manager Stub), User Profile DB, Cache($), Worker(W, Worker API, WS:Worker Stub), Manager, Graphical Monitor가 있다. FE는 Client 요청에 대한 외부 인터페이스를 담당하며, User Profile DB는 사용자와 관련된 정보(Preference)를 저장한다. Cache는 Client의 요청을 처리하며, Worker(Datatype-Specific Distiller)는 데이터에 대한 압축을 수행한다. Manager는 Distiller를 관리하고, Graphical Monitor는 시스템 전체의 상태를 볼 수 있게 해준다.

(그림 2) (a)는 TranSend의 구조를 나타내며, 각 모듈에 대한 자세한 설명은 참고문헌[4, 15]에 기술되어 있다. TranSend에서는 각각의 모듈들(Front End, Cache, Distiller)이 클러스터링 되는 구조로 되어 있다.

(2) 동작 과정

TranSend 구조에서 FE는 부하 분산기(Load Balancer)를 통해 사용자 요청을 받고 이를 캐시로 보낸다. 요청한 데이터가 캐시에 없다면 캐시는 이 데이터를 웹 서버에 요청한다. 캐시가 웹 서버에서 받아온 데이터는 다시 FE로 보내진다. 압축이 필요한 경우, FE는 데이터를 Distiller로 보내어 압축하고, 이를 다시 받아온다. FE는 이 압축된 데이터를 캐시에 저장하고 사용자에게 응답한다.

(그림 2) (b)는 구체적인 동작 과정을 나타내고 이에 대한 자세한 설명은 참고문헌[3]에 기술되어 있다.

2.3 CD-A(CD & All-in-one)

(1) 구조

CD-A[3]는 TranSend 모듈(FE, Cache, Distiller)에서 Distiller를 없애고 Cache에 압축(Distillation) 기능을 추가한 것이다(이하 CD 모듈: Cache & Distiller). 그리고 이 모듈(FE, CD)을 하나의 호스트에 넣고(이하 CD-A 구조: CD & All-in-one[14]) LVS(Linux Virtual Server)[15]를 사용하여 부하 분산을 하는 것이다. Distiller를 없애고 Cache에 압축 기능을 추가한 이유는 복잡한 구조를 단순화하고 불필요한 통신을 줄이기 위해서 이다. 그리고 각 모듈들(FE, CD)을 하나의 호스트에 넣은 이유는 시스템적으로 확장하는 구조를 만들기 위해서이다. 즉, TranSend에서는 새로운 모듈을 추가 시에 동작과정중의 병목 현상을 보이는 모듈을 찾아서 추가해야하는(No Systematic) 반면에, CD-A는 병목에 상관없이 새로운 호스트를 추가하면(Systematic), 그 호스트 내의 모듈(FE, CD) 중에 필요한 모듈이 별도의 설정 없이 상대적으로 많이 사용되는 장점을 가진다.

(그림 3) (a)는 CD-A의 구조를 나타낸다. TranSend에서는 각각의 모듈들(FE, Cache, Distiller)이 클러스터링 되어 있는 반면에 CD-A에서는 각 모듈들(FE, CD)을 하나의 호스트에 통합하고 이러한 호스트들을 클러스터링하는 구조로 되어 있다.

(2) 동작 과정

CD-A 구조에서 FE는 부하 분산기(Load Balancer)를 통해 사용자 요청을 받고 이를 CD로 보낸다. 요청한 데이터가 CD에 없다면 CD는 이 데이터를 웹 서버에 요청한다. 압축이 필요한 경우, CD는 압축을 수행하고 압축된 데이터를 저장한 후 FE를 통해 사용자에게 응답한다.

(그림 3) (b)는 CD-A 구조의 구체적인 동작 과정을 나타내고 이에 대한 자세한 설명은 다음과 같다.

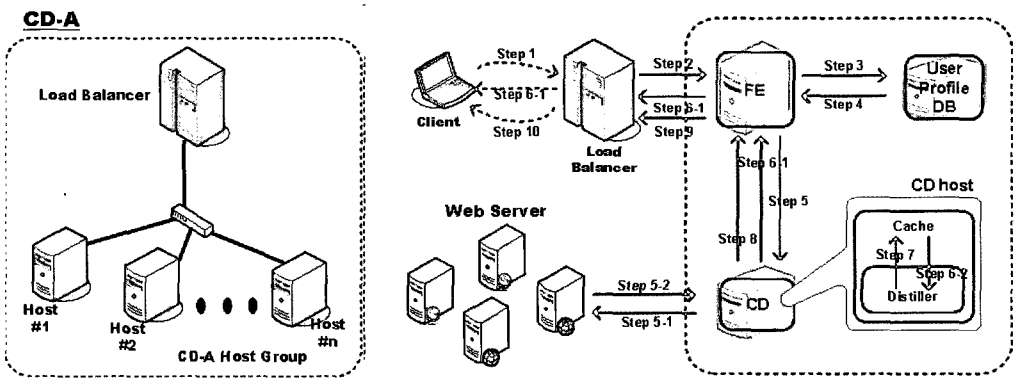
단계 1 : 사용자는 부하 분산기(LB: Load Balancer)에게 데이터를 요청한다.
단계 2 : 부하 분산기는 FE에게 데이터 요청을 보낸다. (이 FE는 부하 분산기에서 스케줄링 방식에 의해 선택된 것이다.)
단계 3 : FE는 User Profile DB에게 사용자 정보(Preference)를 요청한다.
단계 4 : User Profile DB는 사용자 정보를 FE에게 보낸다.
단계 5 : FE는 CD에게 사용자 요청을 보낸다.
  단계 5-1 : CD 안에 요청 데이터가 없다면, 외부 웹서버에 데이터를 요청한다.
  단계 5-2 : 웹서버는 데이터를 CD에 보내고, CD는 이를 저장한다.
단계 6 : CD는 데이터를 확인한다.
  단계 6-1 : 데이터가 압축된 형태라면, 사용자에게 데이터를 보낸다.
  단계 6-2 : 데이터가 압축된 형태가 아니라면, 사용자 정보를 이용하여 데이터를 압축한다.
단계 7 : CD는 압축된 데이터를 저장한다.
단계 8 : CD는 압축된 데이터를 FE에게 보낸다.
단계 9 : FE는 부하 분산기로 압축된 데이터를 보낸다.
단계 10 : 부하 분산기는 사용자 요청에 응답한다.

2.4 접근 방식

본 절에서는 TranSend 및 CD-A 구조가 가지는 문제점을 확장성과 복잡성 측면에서 분석하고, 분석된 문제점을 바탕으로 본 연구의 접근 방식을 소개한다.

(1) TranSend 구조의 문제점

• 확장성(Scalability) : (그림 2) 구조에서 보면 FE, Cache, Distiller는 각각 여러 개의 노드(Node)들로 구성가능하다. 프록시 서버를 확장성 있게 만들려면 노드들을 추가해야하지만 어느 분류(FE 그룹, Cache 그룹, Distiller 그룹)의 노드들을 언제 추가해야 하는지에 대한 시스템적인(Systematic) 방법이 없다. 즉, 실험 결과에 의존하여 특정 모듈 그룹에 병목 현상이 발생하면, 그룹 모듈을 추가하는 방식으로 하



(a) 구조 (b) 동작 과정

(그림 3) CD-A 무선 인터넷 프록시 서버

게 된다. 기존 연구[29]에서는 이를 해결하기위해 All-in-one이라는 구조를 제안하였다.

• 복잡성(Complexity) : TranSend는 FE, Cache, Distiller로 구성되어 FE를 중심으로 서로 간에 통신(Communication)을 하도록 구성되어 있다. 이러한 구조는 FE로 모든 통신이 편중되어 있고 Cache와 Distiller가 분리되어 있어 복잡한 통신을 하는 단점을 가진다.

(2) CD-A 구조의 문제점

• 복잡성(Complexity) : CD-A 구조는 TranSend의 복잡한 구조를 단순화하기 위해 Distiller를 Cache 모듈에 포함시켰다. 그러나 이 구조도 TranSend와 마찬가지로 FE로 모든 통신이 편중되어 있다는 단점을 가진다.

(3) 본 연구의 접근 방식

본 논문에서는 TranSend 및 CD-A 구조의 단점인 복잡성(Complexity)을 단순화(Simplification)하고 시스템적인 확장성을 보장하는 새로운 구조를 제안한다.

3. 새로운 무선 인터넷 프록시 서버

3.1 구조

제안된 새로운 구조에서는 CD-A에서 사용된 기본 모듈에서 FE를 없애고, FE의 기능을 CD에 추가한 것이다. 그리고 이 모듈(하나의 호스트와 동일)을 LVS(Linux Virtual Server)를 사용하여 부하 분산을 하는 것이다. 제안된 구조(Lvs-Cache)에서 FE를 없앤 이유는 다음과 같다. TranSend에서 FE의 주요 기능은 여러 대의 캐시 중에 하나를 선택하는 것이었으나 시스템적인 확장성을 보장하는 CD-A 구조에서는 모듈의 필요성이 없다. 왜냐하면 CD-A 구조는 TranSend를 구성하는 모듈들을 하나의 호스트에 통합하고 이를 클러스터링한 구조이고, FE가 선택할 수 있는 캐시는 자신의 호스트에 있는 것뿐이기 때문에 CD-A 구조에서 FE는 데이터가 지나가는 단순한 통로 역할밖에 하지 않는다.

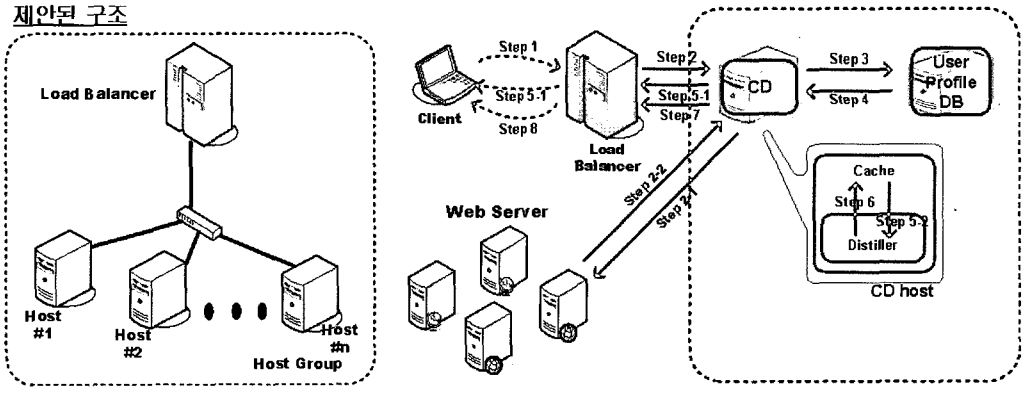
(그림 4) (a)는 제안된 구조를 나타낸다. TranSend에서는 모듈들(FE, Cache, Distiller) 각각이 클러스터링 되어 있고, CD-A에서는 CD-A 호스트가 클러스터링 되어 있는 반면에 제안된 구조에서는 FE와 Distiller를 제외한 Cache가 독립 호스트로서 클러스터링 되어 있다.

3.2 동작 과정

제안된 구조에서는 CD가 부하 분산기(Load Balancer)로부터 사용자 요청을 받아서 처리한다. 요청한 데이터가 CD에 없다면 CD는 이 데이터를 웹 서버에 요청한다. 압축이 필요한 경우, CD는 압축을 수행하고 압축한 데이터를 저장한 후 사용자에게 응답한다.

그림 4(b)는 제안된 구조의 구체적인 동작 과정을 나타내고 이를 자세하게 정리하면 다음과 같다.

단계 1 : 사용자는 부하 분산기(LB: Load Balancer)에게 데이터를 요청한다.
단계 2 : 부하 분산기는 CD에게 데이터 요청을 보낸다. (이 CD는 부하 분산기에서 스케줄링 방식에 의해 선택된 것이다.)
단계 2-1 : CD 안에 요청 데이터가 없다면, 외부 웹서버에 데이터를 요청한다.
단계 2-2 : 웹서버는 데이터를 CD에 보내고, CD는 이를 저장한다.
단계 3 : CD는 User Profile DB에게 사용자 정보(Preference)를 요청한다.
단계 4 : User Profile DB는 사용자 정보를 CD에게 보낸다.
단계 5 : CD는 데이터를 확인한다.
단계 5-1 : 데이터가 압축된 형태라면, 사용자에게 데이터를 보낸다.
단계 5-2 : 데이터가 압축된 형태가 아니라면, 사용자 정보를 이용하여 데이터를 압축한다.
단계 6 : CD는 압축된 데이터를 저장한다.
단계 7 : CD는 압축된 데이터를 부하 분산기 보낸다.
단계 8 : 부하 분산기는 사용자 요청에 응답한다.



(a) 구조 (b) 동작 과정

(그림 4) 제안된 무선 인터넷 프록시 서버

3.3 기존 구조와의 비교

<표 1>은 TranSend와 CD-A를 제안된 시스템(Lvs-Cache)과 구조적으로 비교한 표이다.

〈표 1〉 기존 구조 vs. 제안된 구조(Lvs-Cache)

	TranSend	CD-A	제안된 구조
확장성	없음	있음	있음
구조	LVS-FE-Cache-Distiller	LVS-FE-Cache	LVS-Cache

4. 실험 및 토론

4.1 실험 환경

<표 2>는 실험에 사용된 하드웨어와 소프트웨어를 나타낸다. 무선 인터넷 프록시 서버는 PC 16대로 구성하였고 TranSend 및 제안된 시스템에서 FE의 부하 분산과 All-in-one 시스템의 부하 분산을 위하여 LVS라는 Load Balancer를 사용하였다. Apache Bench라는 프로그램을 Client에서 수행하여 프록시 서버에 영상(이미지)를 요청하는 방식으로 실험하였다. 표에서 Client와 LVS가 Host보다 하드웨어 성능이 좋은 이유는 확장성 실험을 할 때 Client와 LVS에서는 병목이 발생하지 않는 조건에서 프록시 서버 내 호스트들 사이의 확장성을 확인하고자 했기 때문이다.

(그림 5)는 실험에 사용된 구조들의 구성도를 나타낸다. (그림 5) (a)는 제안된 구조를 확장성 측면에서 비교하기 위해, TranSend 기본 구조에 FE를 클러스터링하도록 LVS를 추가하였다.

〈표 2〉 실험용 하드웨어 & 소프트웨어

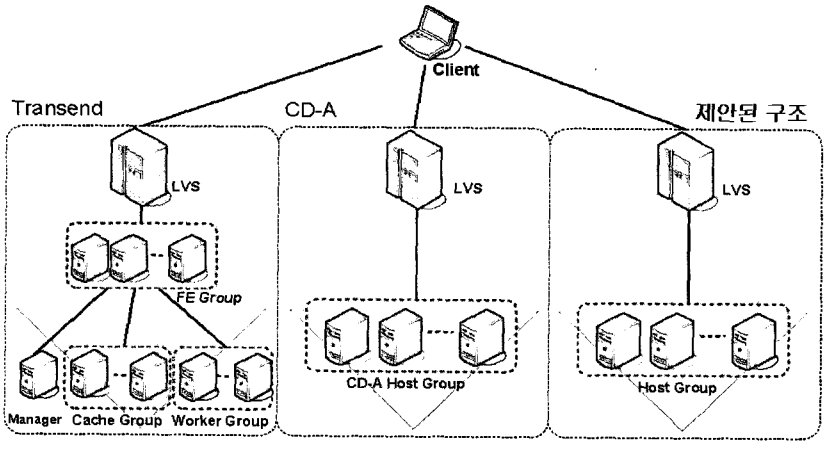
		하드웨어		소프트웨어	개수
		CPU (Hz)	RAM (MB)		
Client		P4 2.26 M	256	AB[16]	2
LVS		P4 2.4 G	512	DR[15]	1
Host	Cache	P2 400 M	256	Squid[17]	16
	Distiller			JPEG-6b[18]	

4.2 실험 방법

<표 3>과 <표 4>는 실험에 사용된 변수 및 각 구조의 실험 방법을 정리한 것이다. 사용자의 요청 개수는 약 200초 동안 프록시 서버가 처리할 수 있는 최대 개수를 사용하였다. 요청 시간을 200초 이상으로 하면 전체적인 실험 결과에는 영향을 미치지 않는 것을 확인하였고, 전체적인 실험 시간을 고려하여 200초로 제한하였다. 사용자의 요청 컨텐츠는 웹에서 가장 많은 사용 빈도를 가지는 JPEG 이미지를 사용하였으며 요청 크기는 300 bytes에서 100 Kbytes 사이의 이미지를 사용하였다. 참고 문헌[19]을 보면 웹에서 가장 많은 사용 빈도를 가지는 이미지는 JPEG이라는 것을 알 수 있다. 요청의 다양화를 위해 Variation Kbytes를 사용하였고, 같은 이름(vari.jpg)으로 1 Kbytes에서 10 Kbytes 까지 10개의 서로 다른 크기의 이미지를 저장하고 사용자가 이를 요청하도록 하였다. 참고 문헌[20]을 보면 웹에서 가장

〈표 3〉 실험에 사용된 변수

사용자의 요청 개수	• 약 200초 동안 프록시가 처리할 수 있는 최대 개수 (예를 들어, 호스트가 1대이고, 요청 이미지가 300 Bytes라면 요청 개수는 100,000개)
요청 이미지	• JPEG
요청 크기	• 300 bytes, 1 K, 10 K, 100 Kbytes, Variation
요청 방식	• 같은 크기 : 100개의 다른 이름(1K00.jpg, 1K01.jpg 등)으로 랜덤(Random)하게 요청(Cache들 사이에서 MD5 Hash를 적용하기 위해) • Variation : 100개의 1K-10K 사이의 이미지를 랜덤하게 요청(Vari01.jpg -> 1K, Vari02.jpg -> 2K 등)
스케줄링 (LVS)	• LVS에서 요청을 분산하는 스케줄링 방식 : RR(Round-Robin) 방식 • TranSend의 경우 FE가 Cache로 요청을 분산하는 방식 : MD5 Hash(캐시간 협동성을 가지도록 하기 위해)
사용자 정보 (Preference)	• 이미지 Quality = 중간
웹 서버	• Cache 서버 자체에 둠(프록시내의 성능 평가에 초점을 맞춤)
병목 (bottleneck)	• 호스트의 CPU 점유율 중 가장 높은 호스트(본 실험에서는 Ethernet이나 System bus 병목은 발생하지 않는다.)



(a) TranSend (b) CD-A (c) 제안된 구조

(그림 5) 실험에 사용된 구성도

〈표 4〉 각 구조의 실험 방법

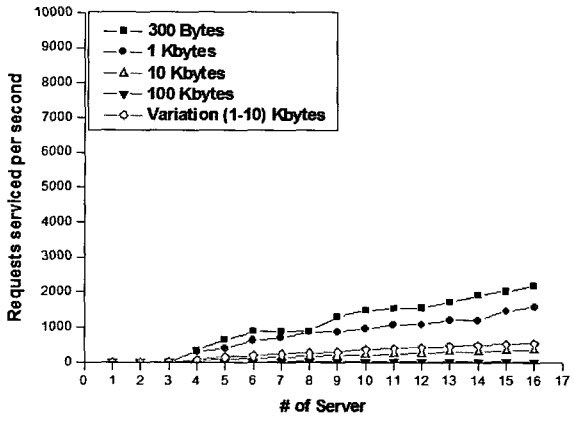
단계	TranSend	CD-A	제안된 구조
1	기본 TranSend 시스템을 구성한다. (Host 1 : Manager/Monitor, Host 2 : FE1, Host 3 : Cache1, Host 4 : Distiller1)	기본 CD-A 시스템을 구성한다. (Host 1)	기본 시스템을 구성한다. (Host 1 : Cache)
2	AB(Apache Bench)를 이용하여 TranSend로 JPEG 이미지를 약 200초 동안 프록시가 처리할 수 있는 최대 개수로 요청한다.	TranSend Step 2와 동일	CD-A Step 2와 동일
3	초당 요청 개수 및 각 Host의 CPU 점유율을 측정하여 병목 Host를 알아낸다.	초당 요청 개수를 측정한다.	CD-A Step 3과 동일
4	병목 Host를 추가한다.	CD-A Host를 추가한다.	Cache 호스트를 추가한다.
5	실험에 사용된 Host 수(16대)만큼 2)-4)를 반복한다.	실험에 사용된 Host 수(16대)만큼 2)-4)를 반복한다.	CD-A Step 4와 동일

(a) 1 Kbytes

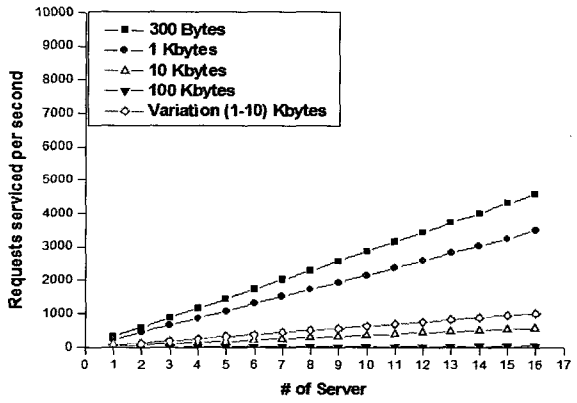
(b) 10 Kbytes

(c) 100 Kbytes

(그림 6) 사용자가 요청한 이미지 샘플



(a) TranSend



(b) CD-A

(그림 7) 호스트 개수에 따른 초당 요청수(TranSend, CD-A)

많은 분포를 가지는 크기는 1-10 Kbytes이다. 이렇게 요청을 하게 되면 같은 동일 서버 내에서 동일 이름으로 서로 다른 크기의 이미지를 요청하는 효과를 가지게 된다. (그림 6)은 사용자가 요청한 이미지 샘플을 나타낸다.

4.3 실험 결과

(1) TranSend & CD-A

(그림 7) (a)는 TranSend 구조에서 이미지 크기를 다르게 요청했을 경우 호스트 개수에 따른 초당 요청수를 나타내고, (그림 7) (b)는 CD-A 구조에서 이미지 크기를 다르게 요청했을 경우 호스트 개수에 따른 초당 요청수를 나타낸다.

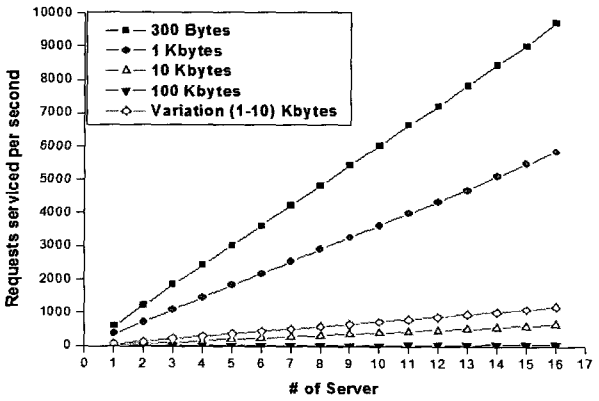
TranSend의 결과를 보면 이미지 크기가 클 경우 호스트 숫자에 비례하여 성능이 향상되나 이미지 크기가 작을 경우 호스트 숫자에 비례(Linear)하여 성능이 향상되지 않음을 알 수 있다. 이는 이미지 크기가 클 경우 Distiller에 집중되어 (데이터를 압축하는 시간이 오래 걸림) 병목이 발생함으로 Distiller만 추가하면 성능이 향상된다. 그러나 이미지 크기가 작으면 병목이 모듈간(FE, Cache, Distiller)에 골고루 분포하게 되어 이를 모두 해결해야만 성능 향상이 발생함을 알 수 있다.

반면 CD-A의 경우 이미지 크기에 상관없이 호스트 숫자에 비례하여 성능이 향상됨을 알 수 있다. 이는 모든 모듈

이 하나의 호스트에 들어있고, 발생된 병목에 따라 필요한 모듈의 CPU 점유율이 상대적으로 높게 배정되는 방식으로 운영되기 때문이다.

(2) 제안된 구조

(그림 8)은 이미지 크기를 다르게 요청했을 경우 호스트 개수에 따른 초당 요청수를 나타낸다. 제안된 구조는 CD-A의 구조적 장점(호스트 개수에 비례하여 성능이 향상됨)을 그대로 가지면서 구조를 더욱 단순화하여 성능이 향상되었음을 알 수 있다.



(그림 8) 호스트 개수에 따른 초당 요청수
(제안된 구조 : Lvs-Cache)

(3) TranSend, CD-A vs. 제안된 구조

<표 5>는 TranSend와 CD-A에 대한 제안된 구조의 평균 성능 향상률을 나타낸 것이다. 제안된 시스템은 TranSend와 CD-A에 비해 각각 평균 196%와 40%의 성능 향상률을 가진다. 작은 크기의 이미지는 큰 크기의 이미지보다 상대적으로 모듈간의 통신에서 많은 시간이 소요됨으로(FE를 통과하는 회수가 많음으로) 제안된 시스템에서 높은 성능 향상률을 가짐을 알 수 있다. 즉, 초당 300개 정도를 처리하는 300 bytes가 초당 2개를 처리하는 100 Kbytes 보다 높은 성능 향상률을 가짐을 의미한다. <표 5>를 보면 이미지 크기가 작아짐에 따라 성능 향상률이 높아짐을 볼 수 있다. 그림 9는 기존 구조와 제안된 구조(Lvs-Cache)를 요청 이미지 크기별로 비교한 것이다.

4.4 토론

제안된 구조의 장점은 다음과 같다.

• 확장성 : 제안된 구조는 기존 TranSend 구조에 비해 시스템적인(Systematic) 확장성을 가진다. 시스템적인 확장성이란 호스트를 추가할 때 병목 모듈에 상관없이 호스트를 추가함에 따라 성능이 높아짐을 의미한다(필요한 모듈이 상대적으로 많이 사용된다). 기존 TranSend 구조는 여러 개의 모듈이 각각의 호스트로 사용되고, 병목은 실험 결과에 의존하여 어느 호스트(모듈)가 병목인지 보고 해당 호스트를 추가해야 하는 단점을 가진다(No Systematic).

• 단순한 구조 : 제안된 시스템이 기존 시스템(TranSend, CD-A)에 비해 성능이 좋아진 이유는 구조적 관점에서 복잡한 구조를 단순한 구조로 바꾼 결과로 설명할 수 있다. 제안된 단순화된 구조는 모듈(FE, Cache, Distiller)간의 통신이 상대적으로 빈번한 작은 크기의 이미지 요청 실험에서 더 좋은 성능 향상률을 보임을 알 수 있다.

• 모듈의 독립성 : 기존 구조로 무선 인터넷 프록시 서버를 구성하게 되면 TranSend에서 사용하는 Front End를 사용해야만 한다. 즉, 무선 인터넷 프록시 서버를 구성하는 모듈이 TranSend의 모듈에 의존하게 된다. 이에 반해 제안된 구조에서는 Front End를 삭제하고, Open Source인 캐시(Squid)와 압축기(JPEG 라이브러리)를 사용함으로써 기존 구조 혹은 모듈에 독립적인 무선 인터넷 프록시 서버의 구성이 가능하다.

제안된 구조의 단점은 다음과 같다.

• Cache : 제안된 구조는 Cache간의 협동성(Cooperation)이 없어서 다른 호스트에 사용자 요청과 동일한 데이터가 Cache되어 있어도 자신의 로컬 Cache에 없다면 매번 실제 웹 서버로 요청하고 이를 자신의 로컬 Cache에 두어서 Cache된 데이터의 중복 합이 커지는 문제가 발생한다. 이 문제를 해결하기 위해서는 LVS에서 스케줄링(부하 분산)시에 해쉬의 적용을 고려해 볼 수 있다. 즉, 해쉬를 적용하게 되면 동일 요청에 동일 캐시를 할당하는 것이 가능하여 전체적으로 캐시의 합이 중복되는 것을 막아준다(Mutually Exclusive).

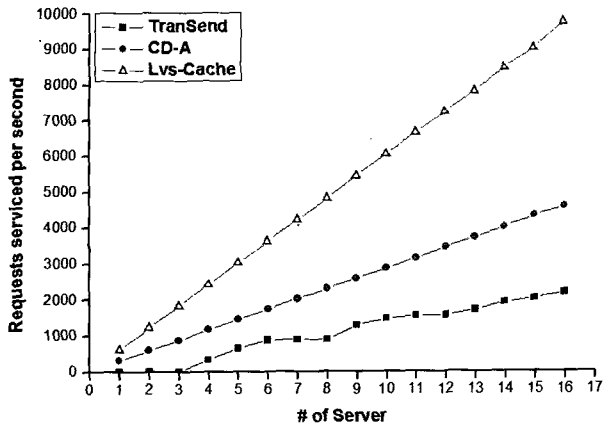
• Hot-Spot : 제안된 구조에서 캐시간 협동성을 위해 해쉬 기반 부하 분산 알고리즘을 사용하게 되면 Hot-Spot(요청 몰림) 문제가 발생하게 된다. Hot-Spot이란 클러스터내의 서버들 중 일부 서버로만 요청이 몰리는 현상으로, 이를 해결하기 위해서는 요청이 몰렸을 때 여유가 있는 다른 서버들이 요청을 같이 처리해주는 알고리즘이 필요하다.

• 압축 시간 : 큰 크기의 이미지를 압축할 때처럼 압축의 비중이 늘어나면 작은 크기의 이미지에 비해 성능 향상률이 감소하게 된다.

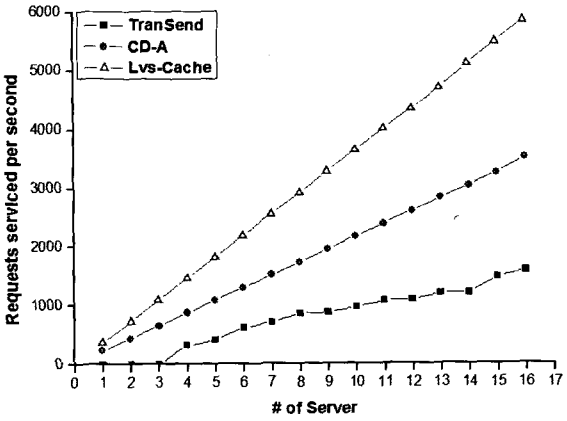
제안된 구조를 실제 무선 인터넷에서 사용할 때 발생 가

〈표 5〉 평균 성능 향상률 (기존 구조 vs. 제안된 구조)

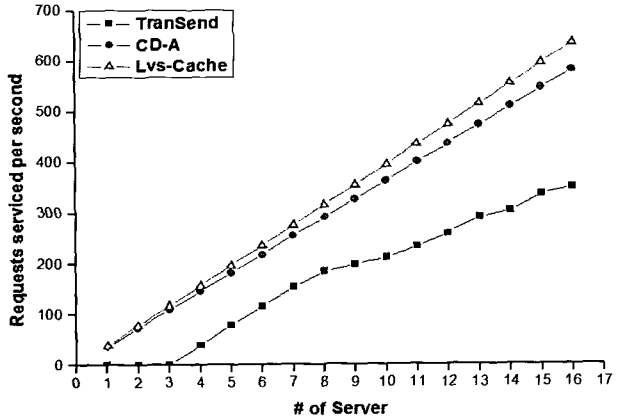
%	300 bytes	1 Kbytes	10 Kbytes	100 Kbytes	Variation	Average
TranSend vs. 제안된 구조	372	287	105	90	126	196
CD-A vs. 제안된 구조	109	67	8	1	15	40



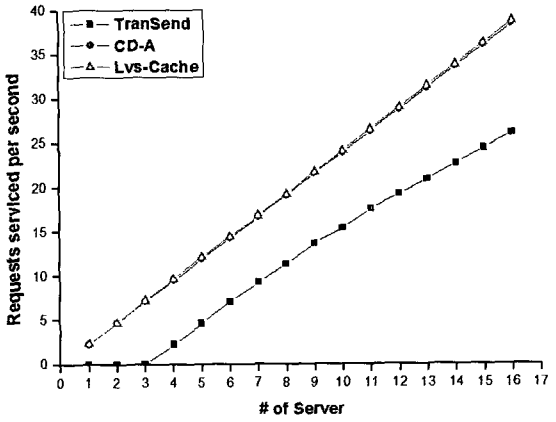
(a) 300 Bytes



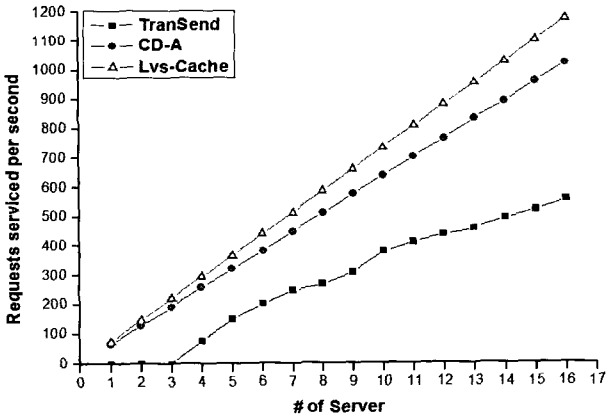
(b) 1 Kbytes



(c) 10 Kbytes



(d) 100 Kbytes



(e) Variation (1-10) Kbytes

(그림 9) 기존 구조(TranSend, CD-A) vs. 제안된 구조(Lvs-Cache) : 요청 이미지 크기별로 비교

능한 문제점들은 다음과 같다.

• 트랜스코딩(Transcoding) : 실제 무선 인터넷에는 클라이언트들이 다양한 기기(핸드폰, PDA 등)를 가지고 있고, 웹 서버에서도 다양한 컨텐츠(이미지, 동영상 등)를 가지고 있다. 원활한 무선 인터넷을 위해서는 무선 인터넷 프록시 서버에서 다양한 컨텐츠를 다양한 클라이언트 기기에 맞게 끔 변환(Transcoding)을 해주어야 한다. 현재 제안된 구조는 이미지를 대상으로 하고 클라이언트 기기가 고정되어 있다고 가정함으로 이에 대한 보완이 필요하다.

• 단일장애점(A single point of failure) : 제안된 구조는 부하 분산기로서 LVS를 사용하고 이를 기반으로 서버들을 클러스터링 한다. 이때, 실제 무선 인터넷 사용시에 사용자

들이 급증하게 되면 LVS의 부하가 가중될 것이고, 만약 LVS에 문제가 생기면(system down) 이 프록시 서버를 사용하는 모든 사용자가 더 이상 무선 인터넷을 사용할 수 없게 된다. 이러한 단일 장애점을 막기 위해 백업(Backup) LVS의 도입을 고려해 볼 수 있다. 즉, LVS에 문제가 발생하면 백업 LVS가 LVS가 하던 일을 대체하는 것이다.

• 보안 : 제안된 구조로 실제 무선 인터넷을 사용하게 되면 유선 인터넷이 가지는 웜(Worm) 등의 보안 위협을 받게 된다. 이러한 무선 인터넷 보안 문제를 해결하기 위해 유선에서 사용하는 IPS (Intrusion Prevention System)를 무선 인터넷 프록시 서버에 적용하는 것을 고려해 볼 수 있다.

5. 결 론

본 논문에서는 무선 인터넷의 근본적인 문제점 중 일부를 해결 할 수 있도록 제안된 TranSend 및 이의 개선 구조인 CD-A 무선 인터넷 프록시 서버의 문제점을 확장성 및 구조의 복잡성 관점에서 지적하고, 이를 해결하는 새로운 구조를 제안하였다. 실험을 통해 제안된 구조가 기존 구조들에 비해 구조 개선 및 성능 향상에 기여함을 확인하였다.

향후 연구 방향을 요약하면 다음과 같다.

• 무선 환경에서의 실험 : 손실률 등을 감안한 실제 무선 환경에서의 실험을 통해 좀더 무선 환경에 적합한 구조나 기능을 가져야 한다.

• 해싱을 이용한 스케줄링의 문제점 보완 : 스케줄링시에 해쉬를 사용하면 동일한 요청에 대해 동일한 캐시가 서비스를 하게 됨으로 전체적인 캐시합을 줄일 수 있다. 그러나 해쉬의 특성상 전체 요청이 일부 캐시로 몰릴 수 있는 가능성이 존재하고, 이는 전체 성능이 일부 캐시에 종속됨을 의미한다. 이를 위해서는 전체 요청이 일부 캐시로 몰리지 않고 균일하게 분포할 수 있는 해쉬 알고리즘이나 그의 응용이 필요하다.

참 고 문 헌

[1] A. Savant, N. Memon and T. Suel, "On the scalability of an image transcoding proxy server", International Conference on Image Processing, to appear, 2003.

[2] A. Feldmann, R. Caceres, F. Douglis, G. Glass and M. Rabinovich, "Performance of web proxy caching in heterogeneous bandwidth environments", In Proceedings of the INFOCOM Conference, 1999.

[3] 곽후근, 정규식, "무선 인터넷 프록시 서버 클러스터 성능 개선", 한국정보과학회논문지 : 정보통신, Vol.32, No.3, pp. 415-426, 2005. 6.

[4] A. Fox, "A Framework For Separating Server Scalability and Availability From Internet Application Functionality", Ph. D. dissertation, U. C. Berkeley, 1998.

[5] M. Liljeberg, H. Helin, M. Kojo and K. Raatikainen, "Enhanced Services for World Wide Web in Mobile WAN Environment", Department of Computer Science, University of Helsinki, Report C-1996-28, 1996.

[6] B. Housel, G. Samaras and D. Lindquist, "WebExpress : A client/intercept based system for optimizing web browsing in a wireless environment", Mobile Networks and Applications, ACM, pp.419-431, 1998.

[7] J. Lee, M. Kim, H. Youn, Y. Hahm and D. Lee, "Class-based proxy server for mobile computers", Proceedings of International Workshops on Parallell Processing, IEEE, pp. 559-566, 2000.

[8] K. Ham, S. Jung, S. Yang, H. Lee and K. Chung, "An Enhanced Proxy Architecture for Efficient Web Browsing over Cellular Networks", Proceedings of the 14th International Conference on Information Networking, pp.5A 4.1-4.5, 2000.

[9] K. Kim, H. Lee and K. Chung, "A Distributed Proxy Server System for Mobile Web Service", Proceedings of the 15th International Conference on Information Networking, IEEE, pp.8A 749-754, 2001.

[10] Anindya Datta and et al., "Proxy-based acceleration of dynamically generated content on the World Wide Web: an approach and implementation", Proceedings of the 2002 ACM SIGMOD international conference on management data, 2002.

[11] A. Maheshwari, A. Sharma, K. Ramamritham and P. Shenoy, "TranSquid : transcoding and caching proxy for heterrogeneous e-commerce environments", Proceedings of 12th International Workshop on RIDE-2EC, IEEE, pp.50-59, 2002.

[12] T. Phan, L. Huang, and C. Dulan, "Integrating mobile wireless devices into the computation grid", Proceedings of the 8th annual international conference on mobile computing and networking", 2002.

[13] Y. Konishi and et al., "Web SHAKE : A fast WWW access method for mobile terminals on tempoary cluster networks", IEEE International Conference on Commjunications, 2002.

[14] 곽후근, 우재용, 정윤재, 김동승, 정규식, "클러스터링 기반의 무선 인터넷 프록시 서버", 한국정보과학회논문지 : 정보통신, Vol.31, No.1, pp.101-111, 2004. 2.

[15] LVS(Linux Vitual Server), http://www.linuxvirtualserver.org.

[16] AB(Apache Bench), http://httpd.apache.org/docs-2.0/programs/ab.html.

[17] Squid Web Proxy Cache, http://www.squid-cache.org.

[18] T. Lane, P. Gladstone and et. al., "The independent jpeg group's jpeg software release 6b.", ftp://ftp.uu.net/graphics/jpeg/jpegsrc.v6b.tar.gz.

[19] T. Kelly and J. Mogul, "Aliasing on the World Wide Web: Prevalence and Performance Implications", Proceedings of the 11th International World Wide Web Conference, pp. 281-292, 2002.

[20] S. Chandra, A. Gehani, C. Ellis and A. Vahdat, "Transcoding Characteristics of Web Images", Proceedings of the SPIE Multimedia Computing and Networking Conference, 2001.

곽 후 근

e-mail : gobarian@q.ssu.ac.kr
1996년 호서대학교 전자공학과(학사)
1998년 숭실대학교 전자공학과(석사)
2006년 숭실대학교 전자공학과(박사)
1998년~2000년 (주)3R부설연구소 주임연구원
2003년~현재 (주)펌킨넷코리아 선임연구원
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 컴퓨터공학(석사)
1990년 미국 University of Southern California 컴퓨터공학(박사)
1998년~1999년 미국 IBM Almaden 연구소 방문연구원
1990년~현재 숭실대학교 정보통신전자공학부 교수
관심분야 : 네트워크 컴퓨팅 및 보안