압축과 캐싱 기능을 향상한 무선 인터넷 프록시 서버 클러스터

곽 후 근, 황 재 훈, 정 규 식

숭실대학교 전자공학과
전화 : 02-821-2757

A Wireless Internet Proxy Server Cluster with Enhanced Distillation and Cashing Functions

Hu Keun Kwak, Jae Hoon Hwang, Kyu Sik Chung

Dept. of Electronics, Soongsil University
E-mail : gobarian@q.ssu.ac.kr

Abstract

A wireless Internet proxy server cluster has to Distillation and Caching functions in order that a user on wireless internet can use existing wired internet service. Distillation function works to distill HTML documents and included images according to the defined preference by a user. When a user requests repeatedly, Caching function decreases response time by reusing original and distilled images or HTML documents.

In this paper, we proposed enhanced distillation and caching functions. We performed experiments using 16 PCs and experimental results show the effectiveness of the proposed system compared to the existing system.

I. 서론

컴퓨터의 사용이 대중화 되고 무선 인터넷 기술이 발전하면서 무선 인터넷의 사용이 증가하고 있으며, 기존의 노트북 외에 PDA, 휴대폰 등의 여러 종류의 무선 단말기가 사용되고 있다. 그러나 무선 인터넷은 특성상 여러 가지 문제들을 가지고 있는데, 대역폭이 낮고, 연결이 자주 끊기며, 단말기마다 컴퓨팅 파워와 화면의 크기가 다른 문제점을 가지고 있다. 이러한 무선 인터넷의 본질적인 문제들을 무선 인터넷 프록시 서버를 이용하여 해결한다. 무선 인터넷 프록시 서버는 압축 및 캐싱 기능을 사용하여 무선 인터넷의 문제들을 개선한다.

본 논문에서는 기존의 압축 및 캐싱 방식을 향상한 무선 인터넷 프록시를 제안하며, 논문의 구성은 다음과 같다. 2장에서는 클러스터링 기반의 무선 프록시 서버 및 Class-based 프록시 서버에 대해서 소개하고, 3장에서는 기존의 압축과 캐싱 방식을 향상한 새로운 알고리즘을 제안한다. 4장에서는 제안된 방법의 실험을, 5장에서는 결론 및 향후 연구 방향을 제시한다.

II. 연구 배경

2.1 클러스터링 기반의 무선 인터넷 프록시

무선 인터넷 프록시는 압축과 캐싱 기능을 사용하며, 대용량 트래픽에 대해 확장성을 고려하여야 한다. TranSend[1] 구조는 이러한 점들을 고려하여 구현된 무선 프록시 서버이다. 본 논문에서는 TranSend를 확장성과 구조적 관점에서 개선한 CD-A(CD & All-in-one) [2]

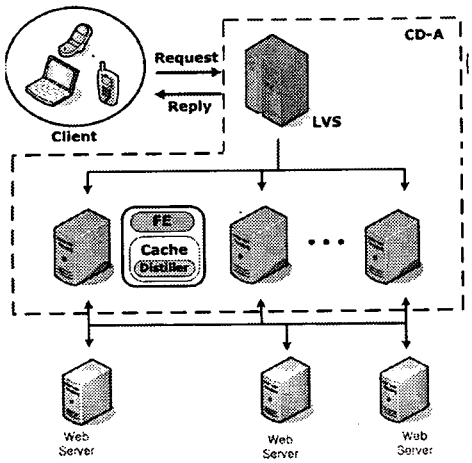

그림 1. CD-A 구조

구조를 사용하였다. 그림 1에서는 CD-A 구조에 대해서 나타내고 있다.

CD-A 구조는 부하 분산을 하는 LVS[3]와 CD-A호스트로 구성되어 있다. 각 CD-A호스트는 FE와 CD 모듈로 구성되어 있으며, FE는 클라이언트 요청에 대한 외부 인터페이스를 담당하고, CD는 클라이언트의 요청을 처리하는 Cache와 데이터를 압축하는 Distiller가 통합된 구조로 이루어져 있다. CD는 기존의 불필요한 통신 구조를 간단하게 줄여서, 압축과 캐싱을 하나의 모듈에서 수행할 수 있는 장점을 가지고 있다. 클라이언트의 요청이 LVS로부터 호스트에 전달되면 호스트 내의 FE가 CD로 전달하며 CD는 요청된 데이터가 존재할 경우 FE에 전달하고, 존재하지 않을 경우 웹서버에 요청하여 얻어 와서 압축한 후 FE에 전달해 클라이언트로 응답하는 방법으로 동작한다.

2.2 Class-Based 프록시 서버[4]

Class-Based 프록시는 HTTP 1.0 과 HTTP 1.1[5]의 프로토콜을 사용하여 구현되었다.

HTTP 1.0에는 Class Mode와 Expert Mode가 있다. 먼저 Class Mode는 사용자가 디스플레이 환경에 따라 클래스를 선택하도록 되어 있어서 별도의 프로토콜 변경 없이 포트 번호를 이용하여 클래스를 선택할 수 있다. 표 1에서는 대표적인 7가지 클래스를 나타내고 있다. 웹 브라우저에서 프록시 서버가 설정되면 IP주소에 따라 각각의 클래스에 대한 포트 번호가 할당되고 사용자는 자신의 환경에 맞는 클래스의 포트 번호로 접속하여, 압축된 데이터를 받음으로써 무선 단말기의 낮은 처리능력의 한계를 보완할 수 있다.

Expert Mode는 각각의 사용자가 자신의 디스플레이 환경에 맞게 컬러/흑백, 색의 수, 최대 압축시간, 이미지

표 1. 클래스별 크기 및 색 구분표

Class	Size(pixels)	Colors
class 1	600 × 480	64K
	640 × 240	64K
class 2	640 × 480	256K
class 3	640 × 240	16gray levels
class 4	640 × 240	monochrome
class 5	240 × 320	256colors
class 6	240 × 320	monochrome
class 7	160 × 160	monochrome

크기의 파라미터들을 설정하여 사용자 각각의 캐시가 할당되며, 서버에 로그인하여 자신의 압축 환경에 맞는 데이터를 받을 수 있다.

HTTP 1.1에서는 별도의 포트번호를 사용하지 않고 무선 단말기가 자신의 CC/PP(Composite Capability/Preference Profiles)프로필을 프록시에 전송하여 프록시 서버가 전송받은 프로필에 따라 사용자의 환경에 맞는 압축된 데이터를 전송하여 처리하도록 한다.

2.3 접근 방식

TranSend 및 기존 프록시 서버와 Class-based 프록시 서버의 캐싱 및 압축 방식의 문제점을 정리하면 표 2와 같다.

표 2. 기존 방식의 문제점

기존방식	문 제 점
TranSend 및 기타 방식	• 사용자가 요청한 하나의 수준으로 압축하고 단말기의 크기에 따른 압축을 하지 못함. • 사용자의 요청이 올 때 새롭게 압축 수준에 맞게 압축하여 보내기 때문에 압축 지연 시간 발생.
Class-based 프록시	• 무선단말기의 크기에 맞는 압축을 하지만 압축 수준(Image Quality)을 고려하지 않음. • 각 모드나 클래스별로 사용자의 요청이 올 때 이미지를 새롭게 압축하여 보내기 때문에 압축 지연 시간 발생

본 논문에서는 기존 방식의 문제점을 토대로 압축 수준(Image Quality)별 압축으로 사용자의 다양한 요청에 대한 서비스 품질(QoS)을 개선하고, 사용자의 요청 가능한 모든 압축 수준으로 데이터를 압축 및 저장하여 사용자의 요청에 대한 응답시간을 단축하는 새로운 방식을 제안한다.

III. 제안된 압축 및 캐싱 방식

그림 2는 2.3절에서 분석된 기존 무선 프록시의 압축 및 캐싱 방식의 문제점을 바탕으로 이를 개선한 방식을 제안한 구조이다. 제안된 압축 및 캐싱 방식은 HTML에 포함된 이미지가 쉽게 바뀌지 않는다는 점을 이용하여, 압축 시간을 단축하고 사용자의 요청에 대

한 서비스 품질(QoS)을 개선하기 위해 제안되었다.

제안된 방식에서 사용자의 요청에 대해 압축 및 저장 기능으로 응답을 수행하는 기존의 방식에, 사용자가 요청 가능한 모든 압축 수준으로 압축 및 저장하는 기능을 추가하였다. 즉, 사용자가 요청 가능한 모든 압축 수준으로 데이터를 압축 및 저장하여 사용자 요청에 대한 응답 시간을 단축하는 것이다.

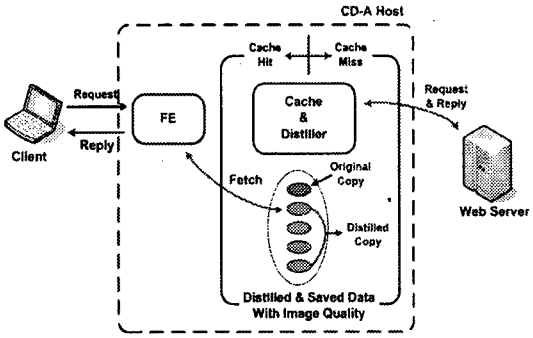

그림 2. 제안된 압축 및 캐싱 알고리즘

기존 방식과 제안된 방식의 동작원리를 비교하면 표 3과 같다. 표 3은 무선 인터넷 프록시 서버 클러스터 상에서 사용자 요청 처리 순서를 나타낸다.

표 3. 기존방식과 제안된 방식의 동작원리 비교

Step	기존 방식(TranSend)	제안된 방식
1	• 사용자가 FE(Front End)에게 이미지를 요청.	기존 방식과 동일
2	• FE는 캐시에 사용자의 압축 수준(Image Quality)을 가지고 이미지를 요청.	
3	• 캐시에 사용자가 요구한 압축 수준에 맞는 이미지가 있다면 FE로 바로 응답.	
4	• 캐시에 사용자가 요구한 압축 수준에 맞는 이미지가 없다면 실제 웹 서버로 원본 이미지를 요청 후 저장.	
5	• 캐시는 원본 이미지를 사용자가 요구한 수준으로 압축 및 저장.	• 캐시는 원본 이미지를 사용자가 요구한 수준 및 요구 가능한 모든 수준으로 압축 · 저장.
6	• 캐시는 사용자가 요청한 압축된 이미지를 FE로 보냄.	기존 방식과 동일
7	• FE는 사용자 요청에 응답	
1	• 사용자가 같은 이미지에 대해 다른 압축수준으로 요청	기존 방식과 동일
2	• 저장된 원본 이미지를 요청한 수준으로 압축 및 저장	•
3	• FE를 통해 사용자 요청에 응답	• 기존의 저장된 이미지를 FE를 통해 응답

제안된 방식은 사용자가 동일 이미지에 대해 다른 압축수준으로 재요청시 기존의 방식보다 압축 및 저장에 소요되는 시간을 단축할 수 있으며, 압축을 위해 필요한 압축기(Distiller)의 수를 줄일 수 있는 장점을 가진다.

Ⅳ. 실험 및 토론

4.1 실험 환경

표 4는 실험에 사용된 하드웨어와 소프트웨어를 나타낸다. LVS 1대를 이용하여 부하분산을 하였고, 프록시 서버는 PC 16대로 구성되었으며, Apache Bench를 이용해 클라이언트에서 이미지를 요청하는 방식으로 실험을 하였다.

표 4. 실험에 사용된 하드웨어 & 소프트웨어

		하드웨어		소프트웨어	개수
		CPU(Hz)	RAM(MB)		
Client		P-III 700M	128	AB	1
LVS		P-IV 2.4G	512	NAT	1
Host	Cache	P-II 400M	256	Squid	1
	Distiller			JPEG-6b	

4.2 실험방법

표 5는 실험에 사용된 변수들을 정리한 것이다. 사용자의 요청에 따라 해당 압축 수준으로 압축 저장 후 응답하는 기존의 방식과 요청에 대해 여러 가지 압축 수준별로 미리 압축 저장해 두었다가 다른 압축 수준으로 요청이 올 때 곧바로 응답하는 방식을 비교 하였다. 그리고 제안된 방식 내에서는 여러 가지 크기의 이미지를 요청할 때 요청 개수에 따른 응답시간을 비교하였다.

표 5. 실험에 사용된 변수

사용자의 요청개수	• 5가지 압축수준에 대하여 1,000, 10,000, 100,000개
요청 이미지	• JPEG[6]
요청크기	• 1K bytes (100개의 다른 이미지)
사용자 정보	• 이미지 Quality = 5단계
웹 서버	• Cache 서버 자체에 둠(프록시내의 성능 평가에 초점을 맞춤)

4.3 실험결과

(1) 기존 방식 vs. 제안된 방식

그림 3은 기존의 압축 및 캐싱 방식과 제안된 압축 및 캐싱 방식의 비교를 시간별로 나타낸다.

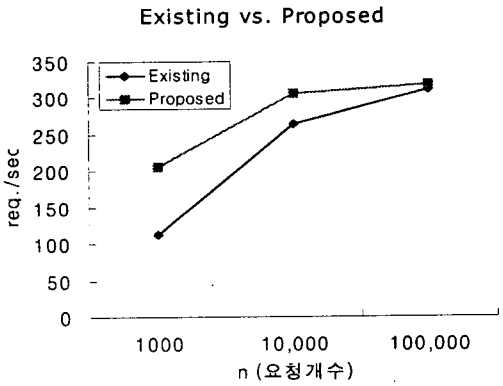

그림 3. 요청 개수에 따른 초당 요청 수 비교

표 6은 기존의 압축 및 캐싱 방식에 대한 제안된 방식의 평균 성능 향상률을 나타낸 것이다. 제안된 압축 및 캐싱 방식은 기존 방식에 비해 평균 1,000개의 이미지를 요청했을 때 45.63% 향상 되었다. 이는 제안된 방식이 기존 방식에 비해 5배의 이미지 수준으로 모두 압축·저장하기 때문에 기존 방식보다 이미지를 많이 공유할 수 있어서 좋은 성능을 내는 것을 볼 수 있다. 그리고 10,000개일 때 13.78%, 100,000개일 때 2.15%의 향상률을 보이는데, 이는 요청 개수가 늘어날수록 제안된 방식의 이미지 저장 및 공유가 기존의 방식과 차이가 줄어들게 되기 때문이다. 그러나 제안된 방식은 이미지를 한 번에 압축하여 저장하므로 기존 방식에서 캐시가 저장하고 있지 않은 이미지에 대한 요청이 왔을 때 원본 이미지를 다른 레벨로 저장해서 요청에 응답할 필요가 없어서 기존 방식보다 우수한 성능을 가지고 있음을 볼 수 있다.

표 6. 제안된 방식의 평균 향상률 비교

요청 개수	1,000	10,000	100,000
기존 방식	112.35	263.71	311.50
제안된 방식	206.65	305.86	318.35
향상률(%)	45.63%	13.78%	2.15%

V. 결론

본 논문에서는 무선 인터넷 프록시 서버의 본질적인 문제점을 해결할 수 있는 압축과 캐싱 방식에서 기존의 압축 및 캐싱 방식이 가지는 문제점을 효율성과 성능 측면에서 지적하고, 이를 개선하는 방식을 제안하였다. 그리고 실험을 통해 제안된 구조가 성능 향상에 기여했음을 확인하였다. 향후 연구 방향으로는 압축과 캐싱에서 압축 수준별 및 디스플레이 크기에 맞게 압축, 저장 하는 방식을 연구하여 압축 저장 시 압축 수준과 함께 디스플레이 크기를 고려하여 저장함으로써 다양한 품질과 크기로 각 무선 단말기에 최적의 응답을 제공하는 방식을 연구한다.

참고문헌

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

[2] H. Kwak, K. Han, and K. Chung, "The Performance Improvement of a Clustering based Wireless Internet Proxy Server," The 31th Spring Conference on KISS, April 2004.

[3] LVS(Linux Virtual Server), http://www.linuxvirtualserver.org

[4] J. Lee, M. Kim, H. Youn, Y. Hahm and D. Lee "class-based proxy server for mobile computers" IEEE, August 2000.

[5] R. Fielding et al., "Hyper Text Protocol - HTTP 1.1." RFC 2616, June 1999.

[6] 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.