투명 캐시에서의 사용자 인증 시스템 구현

김 성 락[†]·구 용 완[††]

요 약

최근의 프록시 서버에서 사용자 인증에 의한 접근제어를 위해서는 각 사용자 브라우저마다 프록시 서버 설정을 해주어야만 하는 불편함이 있다. 본 논문에서 구현한 투명캐시에서의 사용자 인증 기술은 간단히 캐시서버 상에 인증기능 옵션을 설정함으로써 모든 사용자에게는 투명하게 인터넷을 사용할 수 있도록 하였다. 또한 관리자 측면에서는 각 사용자의 트래픽을 감시하고 보안성을 한층 강화하는 효과를 보인다. 그리고 사용자의 인터넷 사용습관을 모니터링 할 수 있어 쇼핑몰과 같은 전자상거래 분야에서 사용자의 성향에 따른 인터넷 전자 고객관계관리(eCRM) 서비스에 활용할 수 있을 것으로 기대된다. 본 기술은 별도로 보안장비의 추가설치 없이 보안이 필수적인 기업 부설 연구소, 전자상거래 웹사이트, 군부대의 인터넷 환경 등에서도 적용할 수 있는 기술이라 하겠다.

Implementation of Client Authentication System on Transparency Cache

Seong-Rak Kim[†]·Young-Wan Koo[††]

ABSTRACT

There are recently a lot of inconvenience because every client should be set to the proxy server on the browser in order to control the access by means of the client authentication in the proxy server. The client authentication technology using the transparency cache in this paper will be transparently used for every user in the internet which option of the authentication function is simply set in the cache server. In addition, the administrator will get the benefit since he can control the traffic of each client and strengthen the security. And also, this system is expected to use in the eCRM deeply related to the tendency of the client in the field of the e-commerce like shopping mall in the internet since the administrator can monitor the pattern of the client using the internet. This technique can be applied to the company affiliated research center, the EC website, and the military where it is essential for the tight security even though there are no additional security devices.

키워드 : 웹 서버(web server), 투명 캐시(transparency cache), 리버스 캐시(reverse cache), 사용자 인증(authentication)

1. 서 론

최근 인터넷 사용자의 급속한 증가와 함께 웹 어플리케이션도 비례하여 많은 발전을 가져왔으나, 웹 페이지를 다운 받는데 몇 초에서 심지어는 몇 분까지 걸리는 경우가 있다. 이를 위해 고성능의 컴퓨터 환경구축이나 네트워크 대역폭의 증설 등으로 이러한 문제를 해결한다는 것은 곤란하다. 이러한 경우의 본질은 고속의 라우터 백본 스위치보다도 인터넷 트래픽 관리 솔루션에 초점을 맞추어 해결해야 할 문제이다.

이에 따라 인터넷 트래픽을 전문적으로 처리해주는 여러 ITM(Internet Traffic Management)제품들이 출시되면서 이에 대한 욕구를 충족시키려 노력하고 있다. 이러한 노력 중에 하나인 웹 캐시는 트래픽의 80%이상을 차지하는 HTTP 트래픽들을 매우 효율적으로 관리하고 처리할 수 있게 해주는 특징으로 인터넷 서비스 업체들의 주목을 받게 되었다.

웹 캐시는 인터넷 가속기(Internet Accelerator)라 한다. 이는 웹 캐시서버 없이 웹 서핑을 했을 때보다 웹 컨텐츠를 빠르게 서비스 해준다는 의미에서 명명되었다. 모든 ITM 제품들이 그렇듯이 웹 캐시 역시 대역폭을 절감해주고, 서버의 부하를 줄여주며, 클라이언트들에게는 더욱 빠른 응답시간을 제공해주는 역할을 한다. 캐시서버를 네트워크에 구현하는 형태는 두 가지로 나누어 볼 수 있다. 하나는 투명 캐시(Transparency Cache)이며, 또 다른 형태는 리버스 캐시(Reverse Cache)이다.

본 연구에서는 투명 캐시에서 사용자 인증을 통한 접근제어를 구현하는데 있다. 기존의 인증 방식은 프록시 서버를 이용하여 웹 사용 허용여부 및 사용자 감시 기능을 제공하였다. 이러한 프록시 인증 방식은 사용자마다 일일이 자신의 웹 브라우저를 설정하여야 하며, 인증 서버의 변경시에는 모든 사용자들의 설정 값을 변경해야 하는 번거로움이 있었다. 투명 캐시는 사용자 브라우저마다 프록시 서버 설정의 필요성을 제거하였지만 인증 체계를 제공하지 않아 인증이 필요한 기관에서는 사용할 수 없다는 단점을 갖는다. 이에 따라 본 연

† 정 회 원 : 오산대학 정보관리과 교수
†† 종신회원 : 수원대학교 컴퓨터과학과 교수
논문접수 : 2001년 7월 26일, 심사완료 : 2001년 11월 21일

구에서 구현한 투명 캐시에서의 사용자 인증 체계는 사용자에게는 투명적인 프록시 구성을 제공하고, 관리자에게는 접근제어를 명확히 함으로써 신뢰할 수 있는 네트워크를 구축할 수 있다.

설계된 사용자 접근제어 모듈은 캐시 엔진과 별도로 작동될 수 있는 외부 모듈로 구현되어 있어 Oracle, DB2, Apache 인증, LDAP 등과 같은 다양한 사용자의 환경에 구축될 수 있는 특징을 갖는다.

2. 관련 연구

캐싱(Caching)이란 캐시에 저장될 수 있는 문서를 웹 캐시에 저장하고 클라이언트의 요청에 의해 캐시된 문서들을 바로 전송하는 기능을 말한다. 대부분의 HTML 문서들과 이미지 파일 등 많은 문서들이 웹 캐시에 저장될 수 있으며, 이러한 문서들이 전체 요청된 문서들의 80%이상을 차지한다. 또한 캐시에 저장된 문서가 클라이언트에 의해 요청되는 Hit율은 평균 60% 정도이다[2]. HTTP는 현재 웹 상에서 가장 많이 사용되어지고 있는 프로토콜 중의 하나이며, 처음 CERN에 근무하고 있었던 Tim Berners-Lee에 의해 제안되었다. 그후 개발된 HTTP/1.0은 인터넷상에서 지금까지 널리 사용되어 왔으나, 기하급수적으로 인터넷의 사용이 늘어나면서 웹의 속도 문제가 제기되어왔다. 이리하여 Roy Eielding에 의해서 처음 HTTP/1.1이 제안되어졌으며, 이것은 End-User에게 좀더 빠른 속도를 제공하며 Persistent Connections, Pipelining, Caching, IP Address Preservation 등을 이용하여 무결성, 안정성 등을 제공할 수 있게되었다[3, 4].

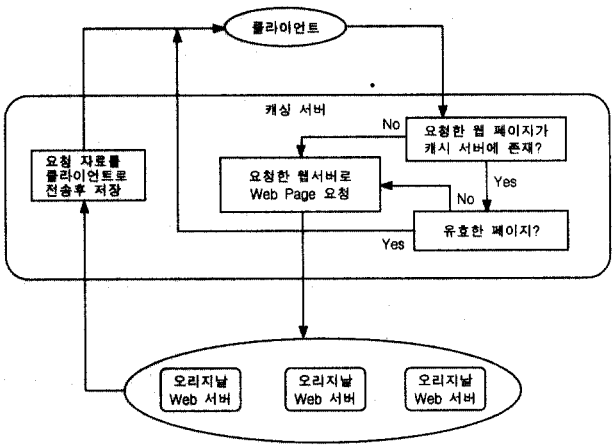
HTTP/1.0 규격에서도 HTTP 프로토콜 동작상의 데이터 캐싱에 대해 다루고 있다. HTTP/1.1에서는 이 기능이 보다 잘 동작하게 하기 위한 많은 요소들이 추가되었는데, 그 목적은 데이터 요청을 위한 요구 메시지의 수를 줄이기 위함이고, 또 다른 경우는 서버가 보내는 응답 메시지의 수를 줄이기 위함이다.

2.1 웹 캐시의 구성 및 동작 메카니즘

웹 캐시의 기본 동작 메카니즘과 구성방법에 따른 일반적인 동작과정은 (그림 1)과 같다[7].

◎ 클라이언트가 www.foo.bar를 접속하려 할 경우

1) 캐시서버에서 요청한 페이지가 있는지 검사
   - 존재할 경우 & 유효한 페이지 : 하드디스크 또는 메모리에서 fetch
     일정시간 내에 요청받은 http 객체를 획득하고, 아직 유효하다고 판정되는 경우는 웹 서버가 아닌 캐시로부터 클라이언트에게 제공한다.
   - 존재하지 않을 경우 & 무효한 페이지 : 요청한 웹 서버(www.foo.bar)로 접속
     유효하다고 판정할 수 없는 경우는 캐시가 서버에게 해당 객체의 유효성 정보를 갱신하고, 객체를 다시 가져와야 할 경우에는 다시 가져와서 저장소에 저장하고 클라이언트에 제공한다.
2) 웹 브라우저로 전송
3) 캐시서버에 저장

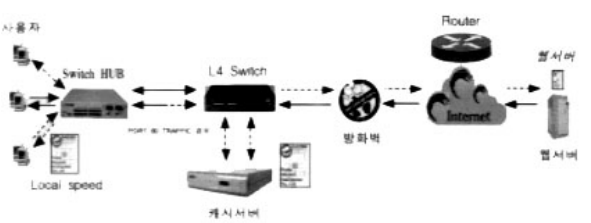


(그림 1) 캐시서버의 흐름도

2.1.1 투명 캐시(Transparency Cache)

투명 캐시는 L4 Switch나 라우터 등에 특수한 장비나 기술을 써서 네트워크 트래픽을 가로채고 이를 처리하게 된다. 투명 캐시와 같은 네트워크 상에 위치한 사용자들은 웹 브라우저에 특별한 설정을 하지 않아도 캐시의 효과를 얻을 수 있게 되는데, 이러한 특징으로 인해 사용자들에게는 캐시가 투명하게 보이게 된다.

투명 캐시는 라우터 레벨과 스위치 레벨의 두 가지 위치에 배치될 수 있는데, 이를 이용해서 여러 대의 캐시를 배치하여 부하분산(load balancing)을 할 수도 있다. 투명 캐시도 동일한 네트워크 상에 있는 많은 사용자들의 요청을 처리할 수 있도록 대부분 네트워크 종단에 위치한다. 아래 (그림 2)는 투명 캐시의 일반적인 구성과 이에 따른 사용자의 요구가 처리되는 과정을 보여주고 있다.



(그림 2) 투명 캐시 구성 및 흐름도

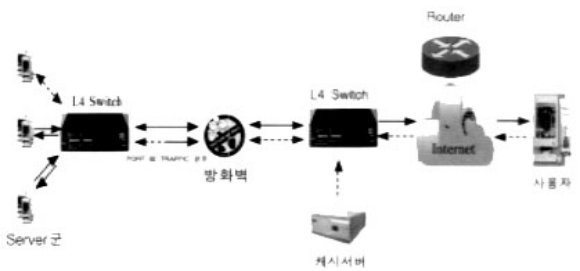
- 첫 번째 사용자가 특정도메인 www.foo.bar를 접속했을 경우

1) Network Switch를 경유(Switch HUB)

2) Layer4 Switch를 경유
3) 캐시서버에 Data가 있는지 검사
4) 없으면 외부의 www.foo.bar에서 Data를 fetch
있으면 캐시서버에서 사용자에게 Data를 서비스
5) 캐시서버에 Data를 저장한 후 사용자에게 전송
6) 이후 다른 사용자가 www.foo.bar에 접속 할 경우 캐시서버에서 Data를 fetch
7) 캐시서버는 주기적으로 www.foo.bar의 update 체크를 통해 최신의 Data를 유지

2.1.2 리버스 캐시(Reverse Cache)

리버스 캐시는 일반 캐시가 불특정 다수의 서버를 대상으로 캐싱하는데 비해 특정한 소수의 서버 데이터만을 캐싱한다는 특성때문에 그 이름을 가지게 되었다. 서버의 부하를 줄이기 위하여 (그림 3)과 같이 서버와 동일한 로컬 네트워크상에 리버스 캐시를 클러스터링(Clustering)을 통해 다수 설치하고, 사용자는 리버스 캐시 클러스터로 요청을 하면 리버스 캐시 클러스터가 캐시된 정보를 사용자에게 제공하는 방식이다. 리버스 캐시는 프록시 캐시나 투명 캐시 같은 클라이언트 측면의 캐시와는 완전히 독립적인 캐시이다. 하지만 리버스 캐시와 프록시/투명 캐시 등을 연동해서 사용한다면 훨씬 더 나은 품질의 서비스 제공이 가능하다. 일반적으로 이와 같은 구성은 대형 포탈 사이트나 인터넷 방송국 등에서 방화벽으로 몰리는 트래픽을 최소화하기 위해 방화벽 앞단에 캐시서버를 두어 사용자에게 높은 응답시간을 주기 위해 구성한다.



(그림 3) 리버스 캐시 구성 및 흐름도

- 사용자가 www.foo.bar를 접속했을 경우
1) 사용자가 라우터를 통해 Layer4 Switch를 경유
2) 캐시서버에서 초기화면을 서비스
3) Data 검색시 캐시서버를 먼저 검색하여 있으면 사용자에게 서비스
없을 경우 방화벽과 L4 스위치를 경유하여 서버군으로부터 데이터를 가져와서 캐시에 저장한 후 사용자에게 서비스
4) 캐시서버는 주기적으로 서버군의 update 체크를 통해 최신의 Data를 유지

2.2 캐시 성능 평가 방법

일반적으로 웹 캐시의 성능을 측정할 수 있는 척도에는 다음과 같은 항목들이 있다.

2.2.1 Hit율(Hit Ratios)

캐시의 성능을 가장 적절히 측정할 수 있는 Hit율은 클라이언트가 요청한 객체가 캐시에 저장되어 있어 바로 클라이언트에게 보내줄 수 있는 비율을 말한다. 캐시의 Hit율은 다음과 같이 계산된다.

$$\frac{Hits}{(Hits + Miss)} \times 100$$

Hits : 요청한 객체가 캐시에 저장되어 있고 유효할 때의 횟수
Miss : 요청한 객체가 캐시에 저장되지 않거나, 유효하지 않을 때의 횟수

Hit율은 웹 캐시가 얼마나 효율적으로 캐싱하는지를 나타내기 때문에 Hit율이 높을수록 훌륭한 캐시라고 볼 수 있다. 특히 Hit율이 좋으면 그 만큼 Miss시의 응답시간을 많이 줄일 수 있어서 전체적인 응답시간도 상당히 향상시킬 수 있다. 하지만 Hit율을 좌우하는 것은 캐시의 대체(Replacement) 알고리즘과 하드디스크의 용량도 관계가 있지만 무엇보다도 캐시를 이용하는 클라이언트들의 요청 패턴에 크게 좌우되기 때문에 객관적인 평가가 매우 힘들다.

2.2.2 단위시간당 최대 처리량(Peak Throughput)

단위 시간당 얼마나 많은 클라이언트의 요청을 처리할 수 있는가는 Hit율과 함께 웹 캐시의 성능을 판가름하는 가장 중요한 척도중 하나이다. 일반적으로 rep/sec(초당 객체 처리수)의 단위로 나타내며, 100Mbps에서 쓰이는 웹 캐시들을 볼 때 50 rep/sec을 처리하는 캐시가 있는가 하면, 950 rep/sec을 처리할 정도로 엄청난 성능을 자랑하는 캐시들도 있다. 하나의 예로 Cache off 3차에서 폴리그래프(polygraph) 벤치마크 프로그램으로 측정한 Compaq-2500인 경우 2398 rep/sec인 반면 Microsoft-C인 경우 571 rep/sec을 나타내었다[5, 6].

2.2.3 응답시간(Response Time)

클라이언트가 요청을 보낸 시간부터 요청한 객체에 대한 응답을 모두 받을 때까지의 시간을 말한다. 응답시간은 Hits일 때와 Miss일 때로 나뉘며, 평균 응답시간은 다음과 같이 계산된다.

$$\frac{(Hits\ Response\ Time \times Hits\ ratio) + (Miss\ Response\ Time \times (100 - Hits\ ratio))}{100}$$

보통 Hits시의 응답시간보다 Miss시의 응답시간이 수십배까지 차이가 나기 때문에, Hit율에 따라 응답시간이 크게 좌우된다. Miss시의 응답시간을 크게 줄일 수 있으면 좋겠지만, Miss시의 응답시간은 캐시의 성능 문제가 아닌 네트워크의

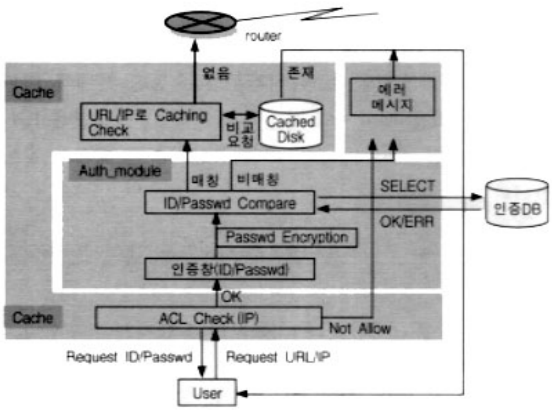
문제이기 때문에 줄이기 힘들다. 또한 Hits시의 응답시간이 캐시 내부 처리와 관련이 깊기 때문에 이를 줄이도록 해야 한다. 많은 캐시들의 응답시간을 비교해 보아도 Miss시의 응답시간은 크게 차이가 없으나 Hits시의 응답시간은 캐시들마다 차이가 매우 다양하게 나타난다.

3. 설 계

본 연구에서 기존 원시 코드(source code)로 사용한 웹 캐시 엔진은 비상업용으로 캐시서버 업체에서 주로 사용하고 있는 Squid를 다루었다[1]. 현재 Squid는 2.5개발버전까지 나와있다. 초기 버전1.0으로부터 시작한 버전은 수많은 기능 추가와 성능향상을 위한 코드가 추가되어 있어 상업용으로 사용하기에도 손색이 없다. Squid가 제공하는 기능들은 HTTP나 FTP의 프록시 및 캐시 기능, SSL의 프록시 기능, ICP와 HTCP, CARP 지원, 그리고 WCCP 지원, SNMP 지원, 투명 캐시 기능, 접근제어기능, DNS 캐시 기능, Persistent Connection 지원을 하고 있다. 세부 모듈들로는 프로토콜을 처리하는 서버 모듈, 클라이언트 접속과 Request 분석/Response 수행을 담당하는 클라이언트 모듈, Request 객체에 대한 메모리, 디스크 관리를 담당하는 저장장치 모듈, 한정된 디스크 공간의 효율적인 사용을 위한 Unlink 모듈, 그밖에 Pass, DNS, Redirect 모듈 등으로 구성되어 있다.

3.1 사용자 인증 모듈

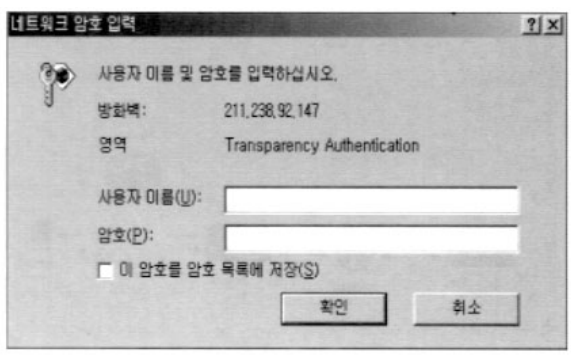
Squid 소스에서 사용자 인증여부를 담당하는 부분은 ACL (Access Control List) 모듈이다. 본 ACL은 사용자의 Request에 대해 Access 허용 여부를 결정하는 모듈이다. ACL에는 파라메타로 요청자의 IP, 접근하려는 Domain Name, 접근 Method 등이 포함된다. 접근 도메인 네임에 대한 IP와 같은 부가적인 정보가 필요할 때는 Lookup 기능을 사용할 수도 있다. (그림 4)는 인증모듈과 웹 캐시 엔진과의 상호 연동 관계와 단계별 수행과정을 나타내었다.



(그림 4) 캐시서버와 인증모듈간의 작업 흐름도

- 인증 모듈의 세부 동작 과정은
  1) 사용자로부터 요청한 도메인과 사용자 IP를 판독
  2) ACL를 검사하여 접속 허용된 사용자인가를 검사(ACL Check)
     - 허용불가 : 접속허용 불가 메시지를 웹 브라우저에 표시
     - 접속허가 : 단계 3
  3) 인증 창을 이용하여 사용자 ID/Passwd를 입력
  4) Base64 암호화 한 후 사용자 DB(또는 패스워드 파일)에서 ID/Passwd 비교
     - 매칭 : 요청 URL 서비스를 위해 단계 5로 이동
     - 비매칭 : 접속 불가 메시지를 웹 브라우저에 표시
  5) 저장된 데이터 존재 유무 판단
     - 존재 : 서비스
     - 비존재 : 오리지날 서버로부터 Fetch하여 서비스

인증 허용기간은 관리자가 설정하기에 따라 다르지만 웹 브라우저 상에 최종 객체가 로드 된 시간과 클릭에 의해 다음번 요청에 의해 최초로 로드되는 객체간에 시간 간격 차이를 계산하여 관리자가 설정한 값을 초과하였을 경우 (그림 5)와 같은 인증 창이 팝업되어 사용자에게 인증을 요구하게 된다. 인증 시도횟수는 기본적으로 3회까지 시도할 수 있으며 3회 이내에 인증이 성공하지 못하면 접근거부 메시지를 보여주고 본 메카니즘은 초기화된다.



(그림 5) 사용자 인증 요구 창

예를 들어 웹 관리자가 인증 유효기간을 10분으로 설정하였을 경우, 해당 네트워크에 속한 한 사용자가 브라우저를 처음 실행하면 인증 창이 팝업되고 미리 등록한 ID/Passwd로 인증을 통과하여 자유로이 웹 서핑을 할 수 있다. 그 후 페이지당 평균 5분을 머무른다면 본 사용자는 아무런 제약 없이 사용할 것이다. 만약 자리를 비우거나 다른 업무로 웹 서핑을 멈춘 시간 간격이 10분을 경과하였다면 다시 웹을 사용할 때에는 인증을 재 요구할 것이다.

3.2 인증 알고리즘

본 연구에서 설계한 ACL 알고리즘은 3개의 함수로 구성

되어 있다. aclMatchAcl은 인증의 유형에 따라 웹 사용 허용 여부를 판단하고, aclMatchProxyAuth는 사용자 ID/Passwd가 정확한지를 검사하며, aclDecodeProxyAuth은 인증 규정 시간 이내의 재접속여부를 판단하고 헤더 정보를 분석하여 헤더 정보가 유효한지를 판단한다.

```
aclMatchAcl(ACL 구조체포인터, ACL 체크리스트 포인터)
{
 switch(ACL->type) { // ACL 유형
          :
 case ACL_URL_PORT :
 case ACL_PROTO :
 case ACL_METHOD :
          :
 case ACL_BROWSER : // 사용자 브라우저 정보 추출
       browser = httpHeaderGetStr(&checklist→header, HDR_
                 USER_AGENT) ;
       break ;
 case ACL_PROXY_AUTH : // Proxy 서버용 인증
       header = httpHeaderGetStr(&checklist->header,
                HDR_PROXY_AUTHORIZATION) ;
       break ;
 case ACL_TRANS_AUTH : // Transparency 서버용 인증
       header = httpHeaderGetStr(&checklist->header,
                HDR_TRANS_AUTHORIZATION) ;
       break ;
          :
 } // end of switch
          :
  switch(aclMatchProxyAuth(인증헤더정보, 인증요구 사용자, 체크
  리스트, ACL 유형)) {
    case 0 : return 0 ; // 패스워드 OK, ACL 접근 거부자 리스트 존재
    case 1 : return 1 ; // 패스워드 OK
    case -2 : return -2 ; // 인증 실패, 재인증 요구
    case -1 : return -1 ; // 패스워드 요구
   }
          :
} // end of aclMatchAcl

aclMatchProxyAuth(인증헤더정보, 인증요구 사용자, 체크리스트,
ACL 유형)
{
 if (!aclDecodeProxyAuth(투명캐시 인증정보 주소 값, 사용자정보 주
 소 값, 패스워드 주소 값))
       return -2 ; // 인증정보 부재, 인증 요구

 if(auth_user) { // 인증요구한 사용자 존재
    if(auth_user->passwd != 1) // 인증요구한 사용자 패스워드 검사
       return -2 ; // 패스워드 오류
    else return 1 ; // 인증 OK
 }
 else
    if(!auth_user) return -1 ; // 인증요구
    else if(체크리스트→사용자 IP = ACL 불허리스트) // ACL 접근 거부
       return 0 ; // 접근 거부자

} // end of aclMatchProxyAuth
```

```
aclDecodeProxyAuth(투명캐시 인증 정보, 사용자ID, 패스워드)
{
 if((현재시간 - 최종 접근시간) <= 인증유효시간 ) { // 인증유효기간
 여부 판단
       last_access_time = current_time ; // 최근 접근시간 갱신
       authen_info=header ; // 인증헤더 갱신
       break ;
 }
 if (authen_info = NULL) return 0 ; // 인증 데이터 없음, 인증 요
 구 필요

 if (password = NULL) return 0 ; // 패스워드 없음
 return 1 ; // 헤더정보 유효

} // aclDecodeProxyAuth
```

접근제어방법은 HTTP 프로토콜의 한 부분이다. HTTP/1.0 규격의 기본 인증체계(Basic Access Authentication)는 사용자 이름과 암호를 묻는 방식에 대한 규정만 있었으나, 새롭게 Proxy-Authentication 필드와 Proxy-Authorization 필드가 추가되어 보안성을 많이 강화하였다[8, 9]. 본 프로토콜의 기본 동작과정을 보면 다음과 같이 클라이언트가 서버에 문서를 요청하면, 이때 서버는 클라이언트 브라우저에게 문서의 길이, 마지막으로 수정된 날짜, 문서의 타입 등의 헤더 정보를 포함하여 함께 클라이언트에게 되돌려준다.

```
<클라이언트 요청>
  GET /documents/ HTTP/1.1
  HOST : www.test.com
```

```
<서버 응답>
  HTTP/1.1 200 OK
  Date : Jun, 11 Mon 2001 12 : 15 : 48 GMT
  Server : Apache/1.3.6 (Unix)
  Last-Modified : Tue, 22 Feb 2000 05 : 45 : 09 GMT
  ETag : "a7dc-68c-3728799e"
  Accept-Ranges : bytes
  Content-Length : 1676
  Content-Type : text/html
```

서버에서 응답한 메시지의 상위에 있는 200은 현 웹 서버의 상태에 대한 코드를 나타내는 것으로 모든 요청을 제대로 수행하여 요청한 문서를 보내주었다는 의미이다. Content-Type은 이 문서가 HTML 형식으로 작성되어져 있다는 것을 의미한다.

이와 반대로 접근제한이 설정되어 있는 웹 서버에 대한 요청은 다음과 같은 응답을 보낼 것이다.

```
HTTP/1.1 401 Authorization Required
Date : Jun, 11 Mon 2001 12 : 30 : 48 GMT
Server : Apache/1.3.6 (Unix) PHP/3.0.9
```

```
WWW-Authenticate : Basic realm = "Transparency Authentication"
Connection : close
Content-Type : text/html
```

위와 같은 메시지는 서버에 의해 요청이 거부되었음을 나타낸다. 클라이언트에서는 서버로부터 이러한 헤드정보를 받으면 **aclMatchAcl** 함수의 ACL_TRANS_AUTH 케이스에 걸려 헤더정보를 읽어올 것을 요구한다. 본 요구를 수신한 클라이언트는 (그림 5)와 같은 인증 요구 창을 띄우고 사용자 아이디와 패스워드 입력을 요구한다.

이렇듯 인증이 필요한 부분은 인증에 대한 부분의 헤더를 입력해 주어야 한다. 예를 들어 사용자 이름이 "intexp"이며 패스워드가 "intexp007"일 경우에는 헤더정보에 인증정보를 추가하여 전송해야 한다.

```
GET /private/ HTTP/1.1
Authorization : Basic aW50ZXhwOmludGV4cDAwNw ==
Host : www.test.com
```

여기서 "Authorization"으로 시작하는 부분에서 "Basic"은 사용되는 인증방법을 의미하며, 시스템에게 인증정보를 보낸다는 뜻이다. "Basic" 뒤에 붙은 문자들은 사용자 아이디와 패스워드를 의미하는 것으로 "username : password"로 구성되어져 Base64로 인코딩되어 전송된다. 헤더가 서버에 재전송되면 **aclDecodeProxyAuthd** 함수에서 이를 관리자가 설정한 시간이내에 재접속한 사용자인가를 판단하여 이를 통과하면 헤더를 디코딩하여 올바른 사용자인가를 판단하게 된다. 인증 규정시간을 초과하게되면 헤더정보는 만료되고 초기 인증 메카니즘으로 되돌아간다. 인증이 통과되면 최종 접속 시간을 갱신하고 OK 신호를 보내어 웹을 사용할 수 있도록 허용하고 사용자에 대한 접속로그를 남겨 사용자의 웹사이트 접속을 추적한다. 본 메카니즘은 단순하면서도 사용자에게 투명성을 제공하며, *웹 서버 관리자에게는 사용자의 웹 사용 모니터링과 인증된 사용자에 대한 접속허가를 자유롭게 설정할 수 있다.

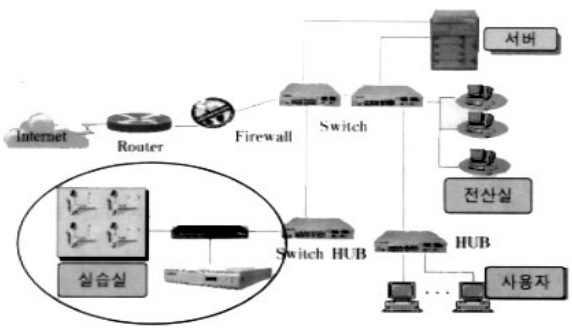
4. 구 현

4.1 실험환경

〈표 1〉 캐시서버 및 L4 스위치 사양

구 분	사 양
프로세서	Intel Pentium III 550MHz
메모리	256 MB ECC
하드디스크	9G
SCSI 콘트롤러	ultra2 (80MB/sec)
LAN 장비	L4 스위치 1대(16포트)

본 실험을 위해 학교에 구축된 기본 네트워크에 실습용 PC 40대가 설치된 학생용 실습실에 L4 스위치와 함께 리눅스 OS 하에서 구동되는 투명 캐시 인증을 지원하는 캐시서버 엔진을 탑재하였다. 인증용 데이터베이스로는 리눅스용 Oracle을 탑재하고, 테스트용 테이블에 실습용 계정과 패스워드 50개 row를 생성하였다. 각 학생들에게 개인별로 계정과 패스워드를 알려주고 다음과 같이 그룹을 정하였다. 그룹 1(1~10)은 IP레벨 접근제어를 설정하여 해당하는 학생의 IP는 접근을 원천적으로 차단토록 하였으며, 그룹 2(11~20)는 연속 인증 실패를 테스트하도록 하였다. 본 그룹은 3회 이상 패스워드가 틀렸을 때 접근 금지 메시지가 브라우저 상에 보이는가를 테스트하기 위함이다. 그룹 3(21~30)은 인증 유효기간을 체크하였다. 이들에게는 인증 유효기간이 5분으로 설정되었기 때문에 인증 유효시간이내에 계속 웹 접속을 하도록 하여 인증을 재차 요구하는가를 체크하였다. 그룹 4(31~40)는 일반 사용자처럼 어떠한 제약사항 없이 웹을 사용하도록 하였다. 이때 인증이 통과되고 키 조작이 없는 시간 간격이 5분을 초과하였을 때 인증을 재요구하는지 등의 모든 조건을 만족하고 웹 페이지에서 보이지 않는 페이지가 없는지를 체크하도록 하였다.



(그림 6) 실험 환경 네트워크 구성도

(그림 6)은 타원 부분이 투명 캐시 실험을 위한 네트워크 구성도이다. 그림에서 볼 수 있듯이 실습실 1곳으로부터 나오는 라인에 L4 스위치를 연결하고 모든 Request가 캐시서버를 향하도록 L4 스위치에 Redirect를 설정하였으며, 캐시서버에서는 ipchains을 이용하여 Port 80에 대한 입력을 캐시 엔진으로 Redirect 되도록 설정하였다. 이와 같은 설정은 실습실에서 일어나는 일련의 인터넷 요청들이 반드시 캐시서버를 거치도록 하기 위함이며, 본 실험결과 각 그룹별로 주어진 작업이 모두 정확히 동작함을 보였다.

4.3 성능평가

실험이 이루어지는 동안 캐시서버에 대한 성능평가결과 단위시간당 최대 처리량이 131 rep/sec을 보였으며, 평균 응답시간은 1936 msec으로 측정되었다.

〈표 2〉 성능 평가표

Throughput	131.86 rep/sec
Response time	1936.08 msec
- misses	3508.39 msec
- hits	503.03 msec
Hit Ratio	54.76 %
Errors	0.00 %

<표 2>에서 알 수 있듯이 캐시에서 Hits가 나지 않고 Miss가 났을 때 원천 서버로부터 데이터를 가져오는데 걸리는 시간이 상대적으로 6~7배 가량 더 소요됨을 알 수 있다. 네트워크 상황이 좋지 않다면 이러한 격차는 더욱 커질 수도 있을 것이다. Hit율은 54.76%를 보였는데, 이러한 수치는 그래프에서도 볼 수 있듯이 요청한 객체들 중 절반 가량이 성공적으로 캐시되었음을 증명한다. <표 3>은 폴리그래프에서 발생시킨 Hit율과 실제 캐시서버에서 발생한 Hit율과의 관계를 보여주고 있다.

〈표 3〉 Hit율 비교

	Hit Ratios
Offered	55.89 %
Measured	54.76 %

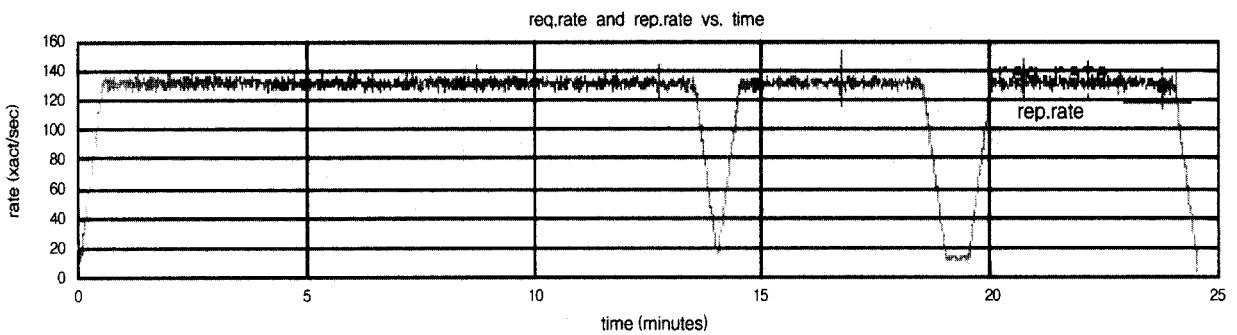
이러한 결과는 Hit를 요구한 거의 모든 오브젝트가 캐시서버에서 Hit되었음을 알 수 있으며, 본 수치는 캐시의 평균 Hit율을 따른다는 것을 알 수 있다.

(그림 7)은 캐시 응답율의 시간대별 추이를 보여주고 있다. 처음 시작할 때 요청 대 응답 비율이 점차 증가하다가 일정한 수준으로 계속 서비스를 한 후 14분과 19분 때에서는 요청이 많이 감소함을 볼 수 있다.

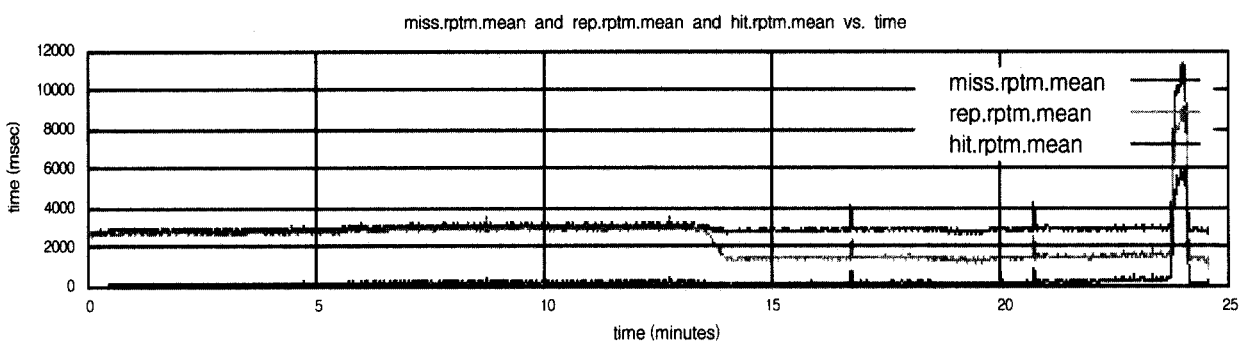
(그림 8)은 캐시의 응답시간 변화를 보여주고 있다. X축 바닥에 거의 깔려있는 선이 객체 Hit시의 응답시간이다. 2200선에 걸쳐있는 파형은 캐시 Miss시 평균 응답시간이다. 13분이 지나면서 초기에는 캐시에 저장된 객체가 없는 관계로 Miss가 많이 발생하다가 점차 캐시에 쌓이면서 캐시 Hit가 많이 발생하여 평균 응답시간이 많이 좋아지고 있음을 알 수 있다. 마지막 23분 무렵에 갑자기 응답시간이 증가한 현상은 캐시 객체를 저장할 공간이 없어 저장되었던 객체를 지우는 과정에서 디스크 오버헤드로 인해 발생한 것으로 분석되었다. 하드디스크 용량을 확보하거나 unlink 데몬 등을 이용하여 이러한 현상을 제거할 수 있다.

5. 결 론

본 연구에서 구축한 투명 캐시에서의 사용자 인증 기술은 사용자의 브라우저 상에서 Proxy 설정을 하지 않고 간단히 웹 캐시서버 상에서 설정을 함으로써 사용자에게는 투명하게 인터넷을 사용할 수 있도록 하고, 동시에 관리자 측면에서는 각 사용자의 트래픽을 감시하고 보안성을 한층 강화하는 효



(그림 7) 캐시 응답율



(그림 8) 캐시 응답시간

과를 얻었다. 본 기술은 별도로 보안장비의 추가설치 없이 보안이 필수적인 기업 부설 연구소나 전자상거래 웹사이트, 전자 고객 관계 관리(eCRM : electronic Customer Relationship Management), 군부대의 인터넷 환경 등에서 적용할 수 있는 기술이라 하겠다.

이는 각 사용자별 방문 사이트와 방문 시간 등의 통계를 로그 분석툴을 이용하여 손쉽게 분석보고서로 작성할 수 있고, 필터링을 통해 특정 사이트를 특정 사용자로부터 차단할 수도 있으며, 또한 전자상거래를 운영하는 곳에서는 특정 페이지의 방문 유형을 분석할 수 있어 고객의 특성에 맞는 서비스를 제공할 수 있는 기술이라 하겠다.

웹 캐시의 성능을 높이기 위한 노력은 끝이 없다. 보다 많은 사용자들의 접속을 처리하기 위해서 Hit율을 최대한 높여 빠른 응답시간의 제공과 대역폭의 절감을 위해서 그리고 확실성과 가용성을 높이기 위해서 많은 연구들이 이루어지고 있고 다양한 기술들이 캐시에 접목되고 있다.

향후 연구과제로는 웹 컨텐츠뿐만 아니라 E-mail 등을 통해 전송되는 컨텐츠에 대해서도 필터링 할 수 있는 기술이 요구된다.

참 고 문 헌

[1] http : //www.squid-cache.org/Doc.

[2] Steven D. Gribble and Eric A. Brewer, System design issues for internet middleware services : Deduction from a large client trace, In Proceedings of the Symposium on Internetworking Systems and Technologies, USENIX, December, 1997.

[3] [RFC2616] "Hypertext Transfer Protocol - HTTP/1.1," p. 176, June, 1999.

[4] [RFC2186] "Internet Cache Protocol (ICP)," version 2", p.9, Sep. 1997.

[5] http : //www.measurement-factory.com/.

[6] http : //polygraph.ircache.net/UserManual.

[7] 최철호 "인터넷 터보 엔진 웹 캐시", 프로그램 세계, 2001.

[8] [RFC2617] HTTP Authentication : Basic and Digest Access Authentication, p.33.

[9] [RFC2069] An Extension to HTTP : Digest Access Authentication, p.18.

김 성 락

e-mail : ksr01@osan.ac.kr

1984년 울산대학교 전자계산학과 졸업(학사)

1989년 한양대학교 산업대학원 전자계산학 전공(석사)

2000년 수원대학교 대학원 전자계산학과 박사과정 수료

1995년~현재 오산대학 정보관리과 조교수

관심분야 : 분산운영체제, 네트워크보안, 인터넷응용, 웹프로그래밍 등

구 용 완

e-mail : ywkoo@cs.suwon.ac.kr

1976년 중앙대학교 전자계산학과 졸업(학사)

1982년 중앙대학교 대학원 전자계산학과 석사과정 졸업(석사)

1988년 중앙대학교 대학원 전자계산학과 박사과정 졸업(박사)

1983년~현재 수원대학교 자연과학대학 컴퓨터과학과 정교수, 수원대학교대학원(일반, 교육, 산업경영)전자계산학과 주임교수, 동 대학교 전자계산소 소장

관심분야 : Operating System을 근간으로한 Distributed System 및 System Software Open System, Multimedia, Real-Time System, Computer Network, Distributed Data Base, Software Engineering, 인터넷응용, 전자상거래 등