Proxy

웹 proxy 서버는 클라이언트의 입장에서 트랜잭션을 수행하는 중개인이다.

HTTP proxy 서버는 웹 서버이기도 하고 웹 클라이언트이기도 하다. 만약 직접 HTTP proxy를 만든다면, HTTP 클라이언트와 HTTP 서버의 양쪽 규칙 모두를 주의 깊게 따라야 한다.

개인 proxy & 공유 proxy

- 개인 proxy: 하나의 클라이언트만을 위한 proxy
- 공유 proxy: 여러 클라이언트가 함께 사용하는 proxy

공용 proxy

대부분의 proxy는 공용이며 공유된 proxy다.

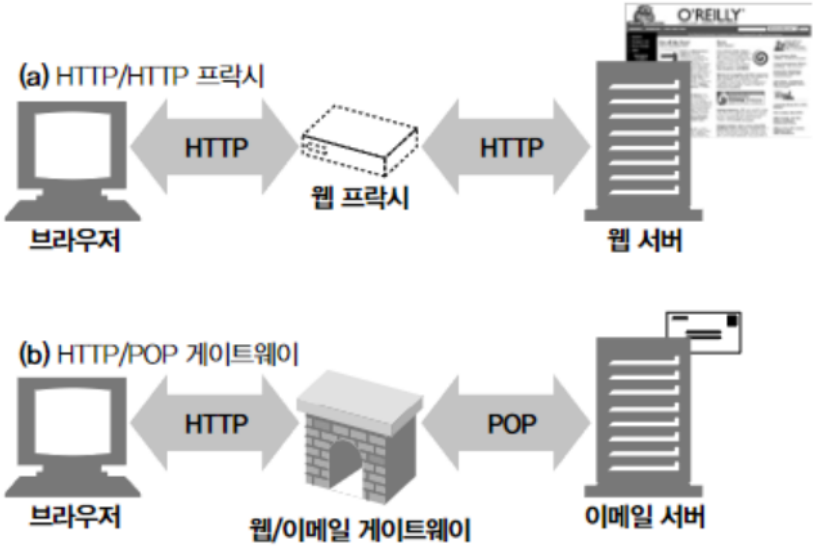
- 중앙 집중형 proxy를 관리하는게 더 비용효율이 높고 쉽다.
- 캐시 proxy 서버와 같은 몇몇 proxy 애플리케이션은 proxy를 이용하는 사용자가 많을수록 유리하다.
    - 여러 사용자들의 공통된 요청에서 이득을 취할 수 있기 때문이다.

개인 proxy

개인 전용 proxy는 흔하지는 않지만 꾸준히 사용되고 있다. (특히 클라이언트 컴퓨터에서 직접 실행되는 형태로)

proxy vs gateway

- proxy
    - 같은 프로토콜을 사용하는 둘 이상의 애플리케이션을 연결한다.
- gateway
    - 서로 다른 프로토콜을 사용하는 둘 이상을 연결한다.
    - 클라이언트와 서버가 서로 다른 프로토콜로 말하더라도 서로 간의 트랜잭션을 완료할 수 있도록 해주는 프로토콜 변환기처럼 동작한다.



실질적으로 proxy와 gateway의 차이점은 모호하다.

브라우저와 서버는 다른 버전의 HTTP를 구현하기 때문에, proxy는 때때로 약간의 프로토콜 변환을 하기도 한다.

상용 proxy 서버는 SSL 보안 프로토콜, SOCKS 방화벽, FTP 접근, 그리고 웹 기반 애플리케이션을 지원하기 위해 게이트웨이 기능을 구현한다.

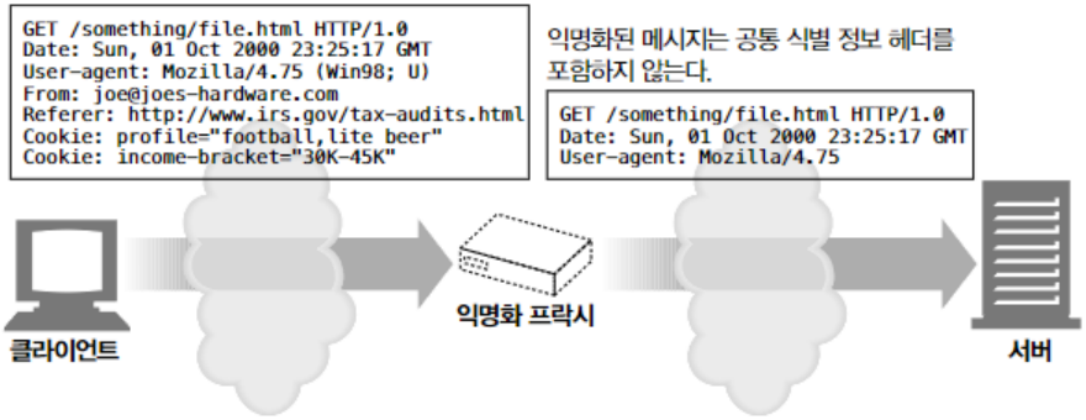
Proxy를 왜 사용하는가?

proxy 서버는 보안을 개선하고, 성능을 높여주며, 비용을 절약한다.

proxy 서버는 모든 HTTP 트래픽을 들여다보고 건드릴 수 있기 때문에, 부가적인 가치를 주는 여러 유용한 웹 서비스를 구현하기 위해 트래픽을 감시하고 수정할 수 있다.

- **어린이 필터**
  - 어린이들에게 교육 사이트를 제공하면서 동시에 성인 콘텐츠를 차단하려고 필터링 proxy를 사용할 수 있다.
- **문서 접근 제어자**
  - proxy 서버는 많은 웹 서버들과 웹 리소스에 대한 단일한 접근 제어 전략을 구현하고 감사 추적(audit trail)을 하기 위해 사용될 수 있다. → 대기업 환경이나 분산된 관료 조직에서 유용
- **보안 방화벽**
  - 네트워크 보안 엔지니어는 종종 보안을 강화하기 위해 proxy 서버를 사용한다.
  - proxy 서버는 조직 안에 들어오거나 나가는 응용 레벨 프로토콜의 흐름을 네트워크의 한 지점에서 통제한다.

    - 바이러스를 제거하는 웹이나 이메일 proxy가 사용할 수 있는, 트래픽을 세심히 살펴볼 수 있는 hook를 제공한다.
- **웹 캐시**
    - proxy cache는 인기 있는 문서의 로컬 사본을 관리하고 해당 문서에 대한 요청이 오면 빠르게 제공하여, 느리고 비싼 인터넷 커뮤니케이션을 줄인다.
- **대리 proxy(Surrogate)**
    - 어떤 proxy들은 웹 서버인 것처럼 위장한다. 때문에 대리 혹은 리버스 proxy로 불리는 이들은 진짜 웹 서버 요청을 받지만 웹 서버와는 달리 요청 받은 콘텐츠의 위치를 찾아내기 위해 다른 서버와 커뮤니케이션을 시작한다.
    - 대리 proxy는 공용 콘텐츠에 대한 느린 웹 서버의 성능을 개선하기 위해 사용될 수 있다. 이런 식으로 사용하는 대리 proxy를 흔히 서버 가속기라고 부른다.
    - 대리 proxy는 콘텐츠 라우팅 기능과 결합되어 주문형 복제 콘텐츠의 분산 네트워크를 만들기 위해 사용될 수 있다.
- **콘텐츠 라우터**
    - proxy 서버는 인터넨 트래픽 조건과 콘텐츠의 종류에 따라 요청을 특정 웹 서버로 유도하는 콘텐츠 라우터로 동작할 수 있다.
    - 콘텐츠 라우터는 사용자들에게 제공할 여러 서비스를 구현하는데 사용할 수 있다.
        - 예를 들어, 사용자나 콘텐츠 제공자가 더 높은 성능을 위해 돈을 지불했다면 요청을 가까운 복제 캐시로 전달할 수 있을 것이다.
        - 또 사용자가 필터링 서비스에 가입했다면 HTTP 요청이 필터링 proxy를 통과하도록 할 수 있을 것이다.
- **트랜스코더**
    - proxy 서버는 콘텐츠를 클라이언트에게 전달하기 전에 본문 포맷을 수정할 수 있다.
        - 이와 같이 데이터의 표현 방식을 자연스럽게 변환하는 것을 트랜스코딩이라고 부른다.
    - 트랜스코딩 proxy는 크기를 줄이기 위해 자신을 거쳐 가는 GIF 이미지를 JPG 이미지로 변환할 수 있다. 또 이미지의 크기를 줄이거나 이미지를 tv에서 볼 수 있게 색 강도를 줄일 수 있다.
    - 마찬가지로, 텍스트 파일은 압축될 수 있고, 인터넷을 이용할 수 있는 무선 호출기와 스마트폰을 위해 작은 텍스트로 줄인 웹페이지를 생성할 수 있다. 필요 시 문서를 바로 외국어 문서로 변환하는 것 또한 가능하다.
- **익명화 proxy(Anonymizer)**
    - 익명화 proxy는 HTTP 메시지에서 신원을 식별할 수 있는 특성들(client IP, From header, Referer Header, Cookie, URI SessionID)을 적극적으로 제거함으로써 개인 정보 보호와 익명성 보장에 기여한다.



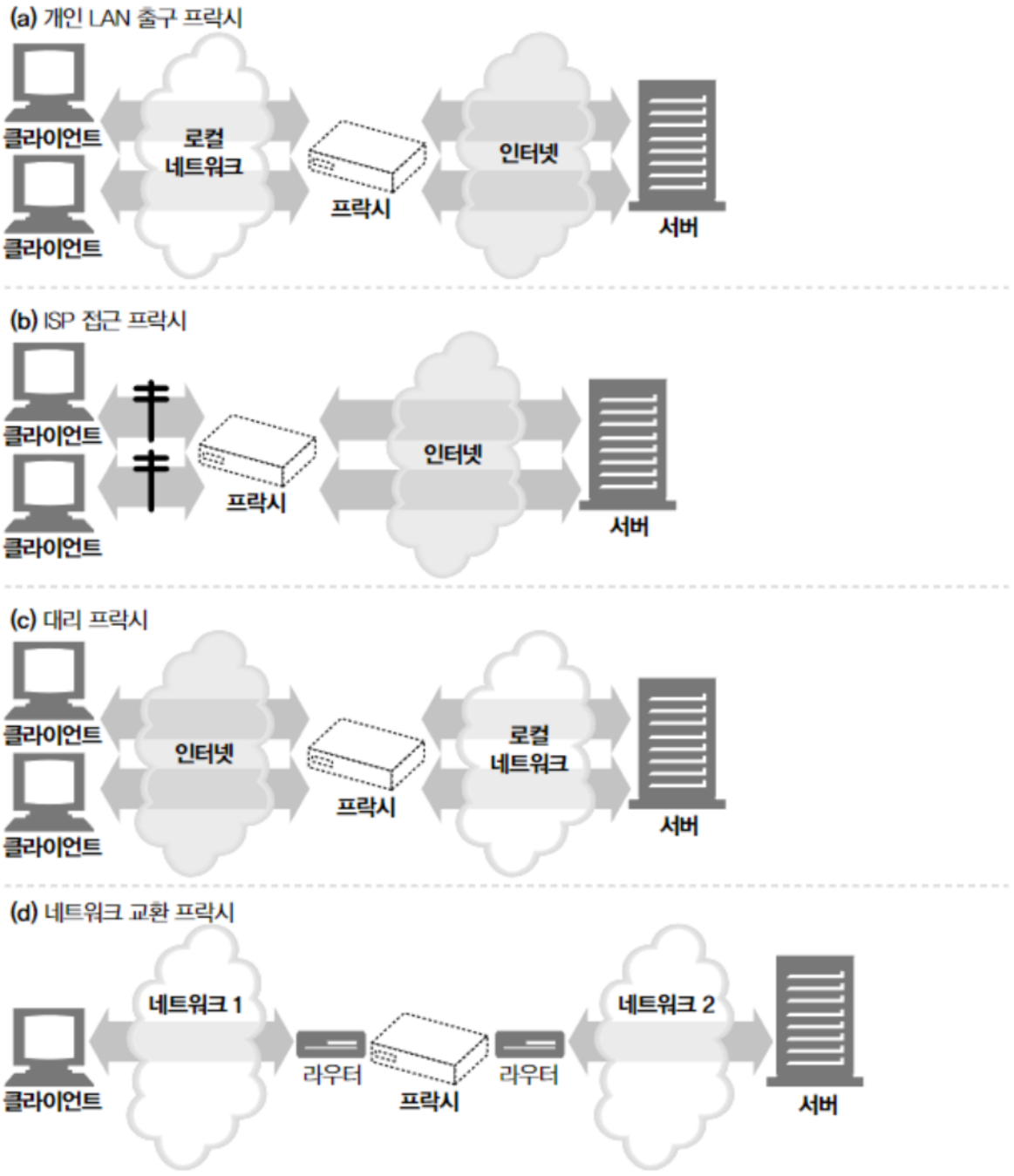
- 위 그림에서, 익명화 proxy는 개인 정보를 보호하기 위해 사용자의 메시지를 다음과 같이 변경한다.
  - User-Agent 헤더에서 사용자의 컴퓨터와 OS의 종류를 제거한다.
  - 사용자의 이메일 주소를 보호하기 위해 From 헤더는 제거된다.
  - 이떤 사이트를 거쳐서 방문했는지 알기 어렵게 하기 위해 Referer 헤더는 제거된다.
  - 프로필과 신원 정보를 없애기 위해 Cookie 헤더는 제거된다.

proxy는 어디에 있는가?

proxy 서버 배치

- 출구(Egress) proxy
  - 로컬 네트워크와 더 큰 인터넷 사이를 오가는 트래픽을 제어하기 위해 proxy를 로컬 네트워크 출구에 박아 넣을 수 있다.
- 접근(입구) proxy
  - 고객으로부터의 모든 요청을 종합적으로 처리하기 위해 proxy는 ISP 접근 지점에 위치하기도 한다.
  - ISP는 사용자들의 다운로드 속도를 개선하고 인터넷 대역폭 비용을 줄이기 위해 캐시 proxy를 사용해 많이 찾는 문서들의 사본을 저장한다.
- 대리 proxy
  - 대리 proxy(reverse proxy)는 네트워크의 가장 끝에 있는 웹 서버들의 바로 앞에 위치하여 웹 서버로 향하는 모든 요청을 처리하고 필요할 때만 웹 서버에게 자원을 요청할 수 있다.
  - 웹 서버에 보안 기능을 추가하거나 빠른 웹 서버 캐시를 느린 웹 서버의 앞에 놓음으로써 성능을 개선할 수도 있다.
  - 대리 proxy는 일반적으로 웹 서버의 이름과 IP 주소로 스스로를 가장하기 때문에, 모든 요청은 서버가 아닌 이 proxy로 가게 된다.

- 네트워크 교환 proxy
    - 캐시를 이용해 인터넷 교차로의 혼잡을 완화하고 트래픽 흐름을 감시하기 위해, 충분한 처리 능력을 갖춘 proxy가 네트워크 사이의 인터넷 피어링 교환 지점들에 놓일 수 있다.

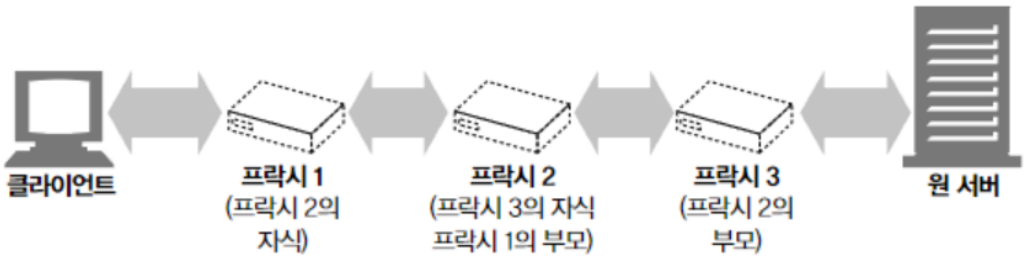


proxy 계층

proxy들은 proxy 계층이라고 불리는 연쇄를 구성할 수 있다.

- proxy 계층에서, 메시지는 최종적으로 원 서버에 도착할 때까지 proxy와 proxy를 거쳐 이동한다 (그 후 다시 proxy들을 거쳐 클라이언트로 돌아온다).

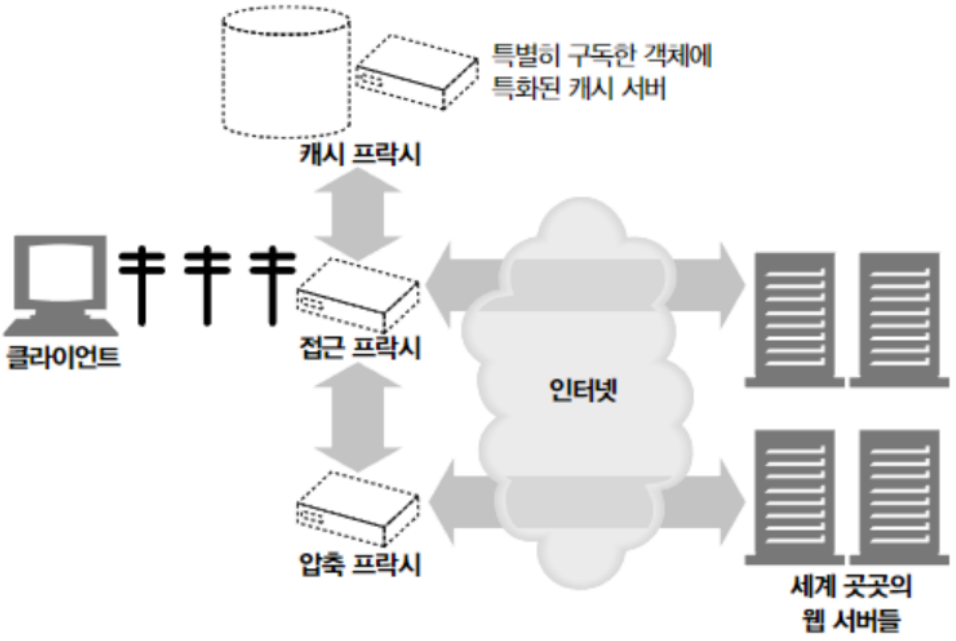
- proxy 계층에서 proxy 서버들은 부모와 자식의 관계를 갖는다.
    - 다음번 인바운드 proxy(서버에 가까운 쪽)를 부모라고 부르고 다음번 아웃바운드 proxy(클라이언트 가까운 쪽)는 자식이라고 부른다.



proxy 계층 콘텐츠 라우팅

proxy 계층은 정적이다. proxy 1은 언제나 메시지를 proxy 2로 보내고, proxy 2는 언제나 메시지를 proxy 3으로 보낸다. 그러나 계층이 반드시 정적이어야 하는 것은 아니다.

proxy 서버는 여러 가지 판단 근거에 의해 메시지를 다양하고 유동적인 proxy 서버와 원 서버들의 집합에게 보낼 수 있다.



예를 들어, 접근 proxy는 상황에 맞게 부모 proxy나 원 서버에게 라우팅한다.

- 요청된 객체가 콘텐츠 분산을 위해 돈을 지불한 웹 서버에 속한 경우, proxy는 요청을 가까운 캐시 서버에게 보내 캐시된 객체를 반환하거나 그럴 수 없을 때는 서버에서 가져오게 할 수 있다.
- 요청이 특정 종류의 이미지에 대한 것인 경우, 접근 proxy는 그 요청을 특화된 압축 proxy에게 보내어 그 proxy가 이미지를 가져와 압축하게 하여 느린 모뎀으로 접속했더라도 빠르게 클라이언트가 다운로드할 수 있게 한다.

동적 부모 선택의 예시

- **부하 균형**
  - 자식 proxy는 부하를 분산하기 위해 현재 부모들의 작업량 수준에 근거하여 부모 proxy를 고른다.
- **지리적 인접성에 근거한 라우팅**
  - 자식 proxy는 원 서버의 지역을 담당하는 부모를 선택할 수도 있다.
- **프로토콜/타입 라우팅**
  - 어떤 자식 proxy는 URI에 근거하여 다른 부모나 원 서버로 라우팅 할 수 있다.
  - 어떤 특정 종류의 URI를 갖고 있는 요청의 경우, 특별한 proxy 서버로 보내져 특별한 프로토콜로 처리될 수도 있다.
- **유료 서비스 가입자를 위한 라우팅**
  - 웹 서비스 운영자가 빠른 서비스를 위해 추가금을 지불했다면, 그들의 URI는 대형 캐시나 성능 개선을 위한 압축 엔진으로 라우팅 될 수 있다.
  - 동적 부모 라우팅 로직은 제품(설정 파일, 스크립트 언어, 동적으로 실행 가능한 플러그인 등)마다 다르게 구현된다.

어떻게 proxy가 트래픽을 처리하는가

- **클라이언트를 수정한다.**
  - 구글 크롬과 마이크로소프트의 브라우저를 포함한 많은 웹 클라이언트들은 수동 혹은 자동 proxy 설정을 지원한다. 만약 클라이언트가 proxy를 사용하도록 설정되어 있다면, 클라이언트는 HTTP 요청을 바로 그리고 의도적으로 원 서버가 아닌 proxy로 보낸다.
- **네트워크를 수정한다.**
  - 클라이언트는 알지도 못하고 간섭도 할 수 없는 상태에서, 네트워크 인프라를 가로채서 웹 트래픽을 proxy로 가도록 조정하는 몇 가지 기법이 있다.
  - 이 가로챔은 일반적으로 HTTP 트래픽을 지켜보고 가로채어 클라이언트 모르게 트래픽을 proxy로 보내는 스위칭 장치와 라우팅 장치를 필요로 한다. 이것을 입터셉트 proxy 라고 부른다.
- **DNS 이름공간을 수정한다.**
  - 웹 서버 앞에 위치하는 proxy 서버인 대리 proxy는 웹 서버의 이름과 IP 주소를 자신이 직접 사용한다. 그래서 모든 요청은 서버 대신 대리 proxy로 간다.
  - 이는 DNS 이름 테이블을 수동으로 편집하거나 사용할 적절한 proxy나 서버를 계산해주는 특별한 동적 DNS 서버를 이용해서 조정될 수 있다.

    - 몇몇 설치본에서는, 실제 서버의 IP 주소와 이름은 변경되고 대리 proxy에게는 이전의 주소와 이름이 주어진다.
- **웹 서버를 수정한다.**
    - 몇몇 웹 서버는 HTTP 리다이렉션 명령(305)을 클라이언트에게 돌려줌으로써 클라이언트의 요청을 proxy로 리다이렉트 하도록 설정할 수 있다.
        - 리다이렉트를 받는 즉시 클라이언트는 proxy와의 트랜잭션을 시작한다.

클라이언트 proxy 설정

수동 설정

proxy를 사용하겠다고 명시적으로 설정한다.

많은 웹 클라이언트가 proxy를 수동으로 설정할 수 있도록 하고 있다. 구글 크롬과 마이크로소프트 인터넷 익스플로러 둘 모두 간편하게 proxy 설정을 할 수 있도록 지원한다.

브라우저 기본 설정

브라우저 벤더나 배포자는 브라우저(혹은 다른 웹 클라이언트)를 소비자에게 전달하기 전에 proxy를 미리 설정해 놓을 수 있다.

proxy 자동 설정(Proxy auto-configuration, PAC)

자바스크립트 proxy 자동 설정(PAC) 파일에 대한 URI를 제공할 수 있다.

클라이언트는 proxy를 써야 하는지, 만약 그렇다면 어떤 proxy 서버를 써야 하는지 판단하기 위해 그 자바스크립트 파일을 가져와서 실행한다.

수동 proxy 설정은 단순하지만 유연하지 못하다. 모든 콘텐츠를 위해 단 하나의 proxy 서버만을 지정할 수 있고, 장애 시 대체 작동에 대한 지원도 없다. 또한 수동 proxy 설정은 큰 조직에서는 관리 문제를 야기한다. 설정된 브라우저가 매우 많다면, 그 모두를 원하는 대로 설정 변경을 하는 것은 어렵거나 불가능하다.

proxy 자동 설정(PAC) 파일은 proxy 설정을 그때끄때 상황에 맞게 계산해주는 작은 자바스크립트 프로그램이기 때문에 proxy 설정에 대한 보다 동적인 해결책이다. 문서에 접근할 때마다, 자바스크립트 함수가 적절한 proxy 서버를 선택한다.

- PAC 파일을 사용하려면, 자바스크립트 PAC 파일의 URI 브라우저에 설정해야 한다. 브라우저는 URI로부터 PAC 파일을 가져와서 매 접근마다 적절한 proxy 서버를 계산하기 위해 자바스크립트 로직을 이용할 것이다.

- PAC 파일은 일반적으로 .pac 확장자를 가지며 MIME 타입은 'application/x-ns-proxy-autoconfig'이다.
- 각 PAC 파일은 반드시 URI에 접근할 때 사용할 적절한 proxy 서버를 계산해주는 `FindProxyForUrl(url, host)`라는 함수를 정의해야 한다.

FindProxyForURL 반환값	설명
DIRECT	프락시 없이 연결이 직접 이루어져야 한다.
PROXY host:port	지정한 프락시를 사용해야 한다.
SOCKS host:port	지정한 SOCKS 서버를 사용해야 한다.

```
function FindProxyForURL(url, host) {
  if (url.substring(0, 5) == "http:") {
    return "PROXY http-proxy.mydomain.com:8080";
  } else if (url.substring(0, 4) == "ftp:") {
    return "PROXY ftp-proxy.mydomain.com:8080";
  } else {
    return "DIRECT";
  }
}
```

WPAD proxy 발견

대부분의 브라우저는 자동설정 파일을 다운받을 수 있는 '설정 서버'를 자동으로 찾아주는, 웹 proxy 자동발견 프로토콜(Web Proxy Autodiscovery Protocol, WPAD)을 제공한다.

WPAD는 여러 발견 메커니즘들의 상승 전략을 이용해 브라우저에게 알맞은 PAC 파일을 자동으로 찾아주는 알고리즘이다. WPAD 프로토콜이 구현된 클라이언트가 하게 될 일은 다음과 같다.

- PAC URI를 찾기 위해 WPAD를 사용한다.
- 주어진 URI에서 PAC 파일을 가져온다.
- proxy 서버를 알아내기 위해 PAC 파일을 실행한다.
- 알아낸 proxy 서버를 이용해서 요청을 처리한다.

WPAD는 올바른 PAC 파일을 알아내기 위해 일련의 리소스 발견 기법을 사용한다.

여러 가지 발견 기법을 사용하게 되는데, 모든 조직이 모든 기법을 사용할 수 있는 것은 아니기 때문이다. WPAD는 성공할 때까지 각 기법을 하나씩 시도해본다.

현재의 WPAD 명세는 다음의 기법을 순서대로 정의한다.

- 동적 호스트 발견 규약(DHCP)
- 서비스 위치 규약(SLP)
- DNS 잘 알려진 호스트 명
- DNS SRV 레코드
- DNS TXT 레코드 안의 서비스 URI

proxy 요청의 미묘한 특징들

proxy URI는 서버 URI와 다르다

웹 서버와 웹 proxy 메서지의 문법은 서로 같지만, 한 가지 예외가 있다. 클라이언트가 proxy 대신 서버로 요청을 보내면 요청의 URI가 달라진다.

클라이언트가 웹 서버로 요청을 보낼 때, 요청 줄은 다음의 예와 같이 스킴, 호스트, 포트번호가 없는 부분 URI를 가진다.

```
GET /index.html HTTP/1.0
User-Agent: SuperBrowserv1.3
```

그러나 클라이언트가 proxy로 요청을 보낼 때, 요청줄은 다음의 예와 같이 완전한 URI를 갖는다.

```
GET http://www.marys-antiques.com/index.html HTTP/1.0
User-Agent: SuperBrowser v1.3
```

원래의 HTTP 설계에서, 클라이언트는 단일한 서버와 직접 대화했다. 가상 호스팅은 아직 존재하지 않았고, proxy에 대한 대비도 없었다.

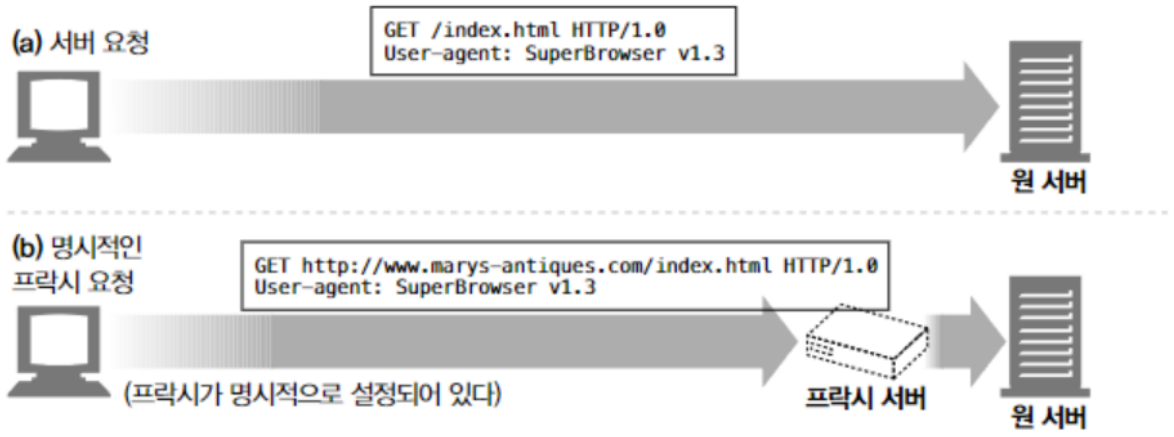
단일 서버는 자신의 호스트 명과 포트번호를 알고 있으므로, 클라이언트는 불필요한 정보 발송을 피하기 위해 스킴과 호스트(그리고 포트번호)가 없는 부분 URI만 보냈다.

proxy가 부상하면서, 부분 URI는 문제가 되었다.

proxy는 목적지 서버와 커넥션을 맺어야 하기 때문에, 그 서버의 이름을 알 필요가 있었다. 그리고 proxy 기반 게이트웨이는 FTP 리소스나 혹은 그 외의 스킴과 연결하기 위해 URI의 스킴을 알 필요가 있었다.

HTTP/1.0은 proxy 요청의 경우 완전한 URI를 요구하는 것으로 이 문제를 해결했지만, 서버 요청의 부분 URI는 여전히 남아있었다.

그래서 우리는 서버로는 부분 URI를, 그리고 proxy로는 완전한 URI를 보낼 필요가 있다. 명시적으로 설정된 클라이언트 proxy 설정의 경우, 클라이언트는 어떻게 요청을 보내야 하는지 알고 있다.



- 클라이언트가 proxy를 사용하지 않도록 설정되어 있다면, 부분 URI를 보낸다.
- 클라이언트가 proxy를 사용하도록 설정되어 있다면, 완전한 URI를 보낸다.

가상 호스팅에서 일어나는 같은 문제

proxy의 '스킴/호스트/포트번호 누락'문제는 가상으로 호스팅 되는 웹 서버가 직면한 것과 같은 문제다.

가상으로 호스팅 되는 웹 서버는 여러 웹 서비스가 같은 물리적 웹 서버를 공유한다.

요청 하나가 부분 URI /index.html로 오면, 가상으로 호스팅 되는 웹 서버는 그 요청이 접근하고자 하는 웹 사이트의 호스트 명을 알 필요가 있다.

이 문제들은 비슷함에도 불구하고, 다음과 같이 각각 다른 방법으로 해결되었다.

- 명시적인 proxy는 요청 메시지가 완전한 URI를 갖도록 함으로써 이 문제를 해결했다.
- 가상으로 호스팅 되는 웹 서버는 호스트와 포트에 대한 정보가 담겨있는 Host 헤더를 요구한다.

인터셉트 proxy는 부분 URI를 받는다

클라이언트가 HTTP를 올바르게 구현했다면, 그들은 명시적으로 설정된 proxy에게는 완전한 URI를 보낼 것이다. 이것으로 문제의 일부분은 해결되지만, 여전히 남은 문제가 있다.

클라이언트는 자신이 proxy와 대화하고 있음을 항상 알고 있는 것은 아니다. 왜냐하면 몇몇 proxy는 클라이언트에게 보이지 않을 수 있기 때문이다.

비록 클라이언트가 proxy를 사용한다고 설정되어 있지 않더라도, 클라이언트의 트래픽은 여전히 대리 proxy나 인터셉트 proxy를 지날 수 있다. 두 가지 경우 모두, 클라이언트는 자신이 웹 서버와 대화하고 있다고 생각하고 완전한 URI를 보내지 않을 것이다.

- 대리 proxy는 원 서버의 호스트 명과 ip주소를 사용해 원 서버를 대신하는 proxy 서버이다.
- 인터셉트 proxy는 네트워크 흐름에서 클라이언트에서 서버로 가는 트래픽을 가로채 캐시된 응답을 돌려주는 등의 일을 하는 proxy 서버이다.
    - 클라이언트에서 서버로 가는 트래픽을 가로채기 때문에, 웹 서버로 보내는 부분 URI를 얻게 될 것이다.

proxy는 proxy 요청과 서버 요청을 모두 다룰 수 있다.

트래픽이 proxy 서버로 redirect될 수 있는 여러 가지 방법이 존재하기 때문에, 다목적 proxy 서버는 요청 메시지의 완전환 URI와 부분 URI를 모두 지원해야 한다.

proxy는 명시적인 proxy 요청에 대해서는 완전한 URI를 사용하고 아니면 부분 URI를 사용해야 하며, 웹 서버 요청의 경우에는 가상 Host 헤더를 사용해야 한다.

완전 URI와 부분 URI를 사용하는 규칙은 다음과 같다.

- 완전한 URI가 주어졌다면, proxy는 그것을 사용해야 한다.
- 부분 URI가 주어졌고 Host 헤더가 있다면, Host 헤더를 이용해 원 서버의 이름과 포트 번호를 알아내야 한다.
- 부분 URI가 주어졌으나 Host 헤더가 없다면, 다음의 방법으로 원 서버를 알아내야 한다.
    - proxy가 원 서버를 대신하는 대리 proxy라면, proxy에 실제 서버의 주소와 포트 번호가 설정되어 있을 수 있다.
    - 이전에 어떤 인터셉트 proxy가 가로챘던 트래픽을 받았고, 그 인터셉트 proxy가 원 ip주소와 포트번호를 사용할 수 있도록 해두었다면, 그 ip 주소와 포트번호를 사용할 수 있다.
    - 모두 실패했다면, proxy는 원 서버를 알아낼 수 있는 충분한 정보를 갖고 있지 못한 것이므로 반드시 에러 메시지(보통 사용자에게 Host 헤더를 지원하는 현대적인 웹브라우저로 업그레이드를 하라는 것이다)를 반환해야 한다.

전송 중 URI 변경

proxy 서버는 요청 URI의 변경에 매우 신경을 써야 한다. 무해해 보이는 사소한 URI변경이라도 다운스트림 서버와 상호운용성 문제를 일으킬 수 있다.

특히 몇몇 proxy는 URI를 다음 홉으로 보내기 전에 표준 형식으로 '정규화'하는 것으로 알려져 있다.

URI에서 기본 HTTP 포트를 명시적인 ':80'로 변경하는 것이나 잘못 사용한 예약된 글자를 올바르게 이스케이프하여 교체하는 것과 같은 무해해 보이는 변형이라 할지라도, 상호운용성 문제를 일으킬 수 있다.

일반적으로 proxy 서버는 가능한 한 관대하도록 애써야 한다. 그들은 프로토콜을 엄격하게 준수하도록 강제하는 '프로토콜 경찰'처럼 되려고 해서는 안 된다. 이는 기존에 잘 동작하던 기능들을 심각하게 망가뜨리는 결과를 수반할 수 있기 때문이다.

특히 HTTP 명세는 일반적인 인터셉트 proxy가 URI를 전달할 때 절대 경로를 고쳐 쓰는 것을 금지한다. 유일한 예외는 빈 경로를 '/'로 교체하는 것 뿐이다.