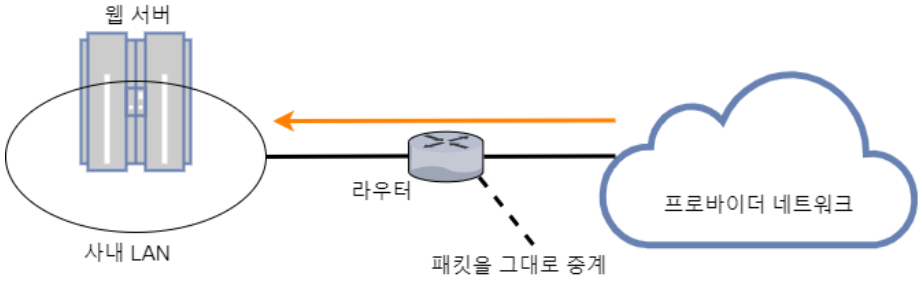
Chapter 05

방화벽과 캐시 서버

1. 웹 서버의 설치 장소

사내에 웹 서버를 설치하는 경우

- 인터넷을 빠져나와서 서버에 도착할 때까지의 여정은 서버의 설치 장소에 따라 다르다.
- 가장 간단한 것은 그림과 같이 사내의 LAN에 서버를 설치하고, 인터넷에서 직접 액세스하는 경우이다.

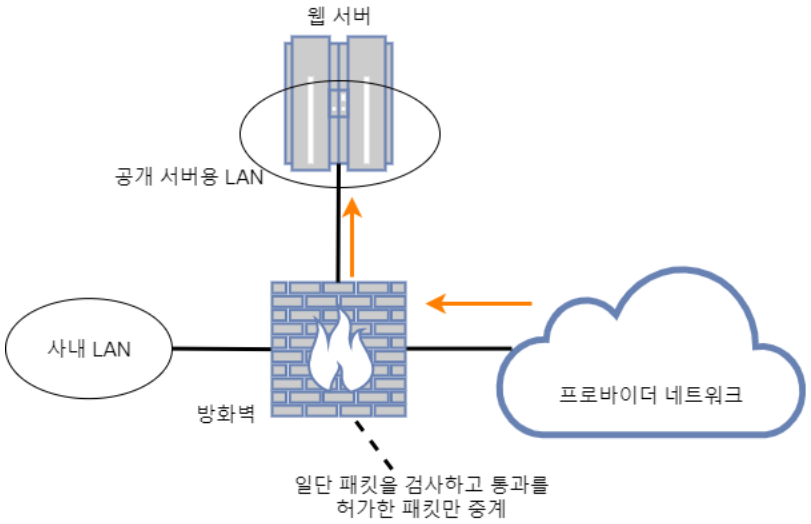


- 이전에는 이런 형태로 서버를 설치하는 경우가 많았지만, 현재 이 방법은 주류에서 밀려났다.
  - IP 주소의 부족.
  - 보안상의 이유.

1. 웹 서버의 설치 장소

사내에 웹 서버를 설치하는 경우

- 지금은 그림과 같이 방화벽을 두는 방법이 일반화되었다.

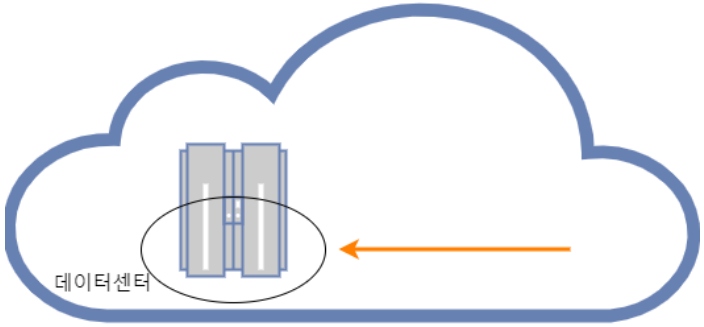


- 방화벽은 관문의 역할을 하여 특정 서버에서 동작하는 특정 애플리케이션에 접근하는 패킷만 통과시키고, 그 외의 패킷을 차단하는 역할을 한다.
- 이를 통해 애플리케이션에 보안 구멍이 있어도 위험성이 적어진다.

1. 웹 서버의 설치 장소

데이터센터에 웹 서버를 설치하는 경우

- 웹서버는 회사 안에만 설치하는 것이 아니다.
- 그림과 같이 프로바이더 등이 운영하는 데이터센터라는 시설에 서버를 가지고 들어가서 설치하거나 프로바이더가 소유하는 서버를 빌려쓰는 형태로 운영하는 경우도 있다.
- 데이터센터는 인터넷의 중심 부분에 고속 회선으로 접속되었으므로 여기에 서버를 설치하면 고속으로 액세스할 수 있다.
- 이것은 서버에 대한 액세스가 증가했을 때 효과적이라고 할 수 있다.



- 또한 데이터센터는 서버의 설치 장소를 제공할 뿐만 아니라 기기의 가동상태 감시, 방화벽의 설치 운영, 부정 침입 감시라는 부가 서비스를 제공하는 경우가 있으며, 안전성도 높다.

2. 방화벽의 원리와 동작

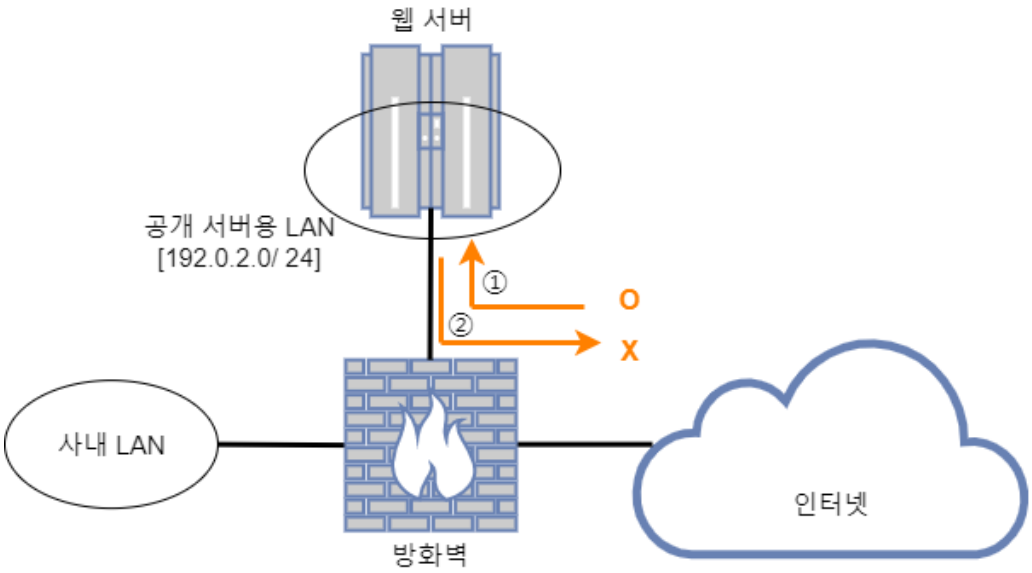
패킷 필터링형이 주류

- 서버의 설치 장소와 관계없이 지금은 바로 앞에 방화벽이 있는 것이 보통이다.
- 방화벽의 기본이 되는 개념은 앞에서 설명했듯이 특정 서버와 해당 서버 안의 특정 애플리케이션에 접근하는 패킷만 통과시키고, 그 외의 패킷을 차단한다.
- 그러나 특정 서버나 특정 애플리케이션의 패킷만 통과시킨다고 간단히 말했지만, 네트워크에는 다양한 종류의 패킷이 많이 흐른다.
- 그러므로 이 중에서 통과시킬 패킷과 차단할 패킷을 선별하는 것은 간단한 일이 아니기 때문에 지금까지 다양한 방법이 고안되었다.
- 성능 가격 사용 편의성 등의 이유로 지금은 패킷 필터링형이 가장 많이 보급되었다.

2. 방화벽의 원리와 동작

패킷 필터링의 조건 설정

- 그림과 같이 공개용 서버를 설치하는 LAN과 사내 LAN이 분리되어 있고, 웹 서버는 공개 서버용 LAN에 접속되어 있다고 가정하자.
- 그리고 인터넷에서 웹 서버에 대한 접속을 허가하지 않으면(그림 ①) 웹 서버에서 인터넷의 접속을 금지하도록 패킷을 차단한다(그림 ②).



패킷 필터링의 조건 설정

조건					통과/차단
수신처 IP 주소	수신처 포트 번호	송신처 IP 주소	송신처 포트 번호	TCP 컨트롤 비트	
192.0.2.0/ 24	80	-	-	-	통과
-	-	192.0.2.0/ 24	80	SYN=1 ACK=0	차단
-	-	192.0.2.0/ 24	80	-	통과
-	-	-	-	-	차단

패킷 필터링의 조건 설정

- 이전에는 웹 서버에서 인터넷측에 대한 접속을 금지하는 예가 적었다.
- 하지만 지금은 서버에 기생하며 여기에서 다른 서버에 감염되어 가는 악성 소프트웨어가 있으므로 감염을 막기 위해 웹 서버에서 인터넷측에 접속하는 것을 금지하고 있다.
- 이러한 상황에서 패킷 필터링의 조건을 어떻게 설정할까?
- 패킷 필터링의 조건을 설정할 때는 먼저 패킷의 흐름에 착안한다.
- 그리고 수신처 IP 주소와 송신처 IP 주소에 따라 시점과 종점을 판단한다.
- 그림의 ①의 예라면 인터넷에서 웹 서버를 향해 패킷이 흐른다.
- 인터넷에서 보내오는 패킷은 시점을 지정할 수 없지만 흐름의 종점은 웹 서버가 되므로 이것을 조건으로 설정하고 조건에 해당하는 패킷만 통과시킨다(표의 1행).
- 이렇게 해서 인터넷측에서 웹 서버로 흘러가는 패킷은 방화벽을 통과하지만, 이것만으로는 액세스 동작이 제대로 작동하지 않는다.
- 패킷을 받으면 정확하게 도착했는지를 송신측에 알리는 수신 확인 응답의 구조가 작용하므로 웹 서버에서 인터넷측으로 흐르는 패킷도 있기 때문이다.
- 웹 서버에서 클라이언트에 보내는 데이터가 있고 이 패킷도 웹 서버에서 인터넷측으로 흘러간 후 그곳에서 시점(송신처 IP 주소)이 웹 서버의 주소에 일치하는 패킷도 통과한다(표의 3행).

패킷 필터링의 조건 설정

- 이것 뿐이라면 인터넷과 웹 서버 사이를 흐르는 패킷은 전부 통과해서 위험한 상태가 된다.
- 그러므로 불필요한 접근, 이 예에서는 웹 이외의 애플리케이션의 패킷을 전부 차단하는 쪽이 좋다.
- 이와 같이 애플리케이션을 한정할 때는 TCP 헤더나 UDP 헤더에 기록되어 있는 포트 번호를 조건으로 추가한다.
- 웹 서버의 포트 번호는 80번으로 결정되어 있으므로, 전술한 수신처 IP 주소 및 송신처 IP 주소에 수신처 포트 번호가 80번인 경우에 대한 조건도 추가한다(표의 1행).
- 또한 송신처 IP 주소가 웹 서버의 주소와 일치하고 송신처 포트 번호가 80번인 패킷도 통과하도록 설정한다(표의 3행).
- 그리고 웹 서버 이외의 애플리케이션에 대한 접근을 허가할 경우에는 이 애플리케이션의 포트 번호를 설정하여 이것이 통과하도록 한다.

2. 방화벽의 원리와 동작

컨트롤 비트

- 이렇게 해서 애플리케이션도 지정했지만 아직 조건이 부족하다.
- 웹 서버에서 인터넷측에 접속하는 동작을 정지시킬 수 없기 때문이다.
- 웹의 동작은 TCP 프로토콜을 사용하여 양방향으로 패킷이 흐르므로 단순히 웹 서버에서 인터넷으로 흐르는 패킷을 정지시키면 인터넷에서 웹 서버에 접속하는 동작도 정지된다.
- 패킷이 흐르는 방향이 아니라 접속 방향을 판단하여 정지시켜야 하는데, 여기에서 도움이 되는 것이 TCP 헤더에 있는 컨트롤 비트이다.
- TCP는 최초에 행하는 접속 단계의 동작에서 3개의 패킷이 흐르는데, 최초의 패킷만 TCP 컨트롤 비트의 SYN이라는 비트가 1로 되고, ACK라는 비트가 0이 된다.
- 다른 패킷에서 같은 값을 취하는 경우는 없으므로 이 값을 조사해서 최초의 패킷과 두 번째 이후의 패킷을 판별할 수 있다.

컨트롤 비트

- 최초의 패킷이 웹 서버 측에서 인터넷측으로 흘러갈 경우 이것을 차단하도록 설정하면 어떻게 될까(표의 2행)?
- 이것을 차단하면 상대로부터 두 번째 패킷이 돌아오는 경우가 없으므로 당연히 TCP의 접속 동작은 실패로 끝난다.
- 이렇게 해서 웹 서버에서 인터넷에 접속하는 동작을 정지시킬 수 있다.
- 인터넷측에서 웹 서버에 액세스하는 패킷은 어떻게 될까?
- 인터넷측에서 웹 서버에 액세스할 때 최초의 패킷은 수신처가 웹 서버를 나타내고, [표]에서 첫 번째 행의 조건에 해당하므로 패킷 필터링을 통과한다.
- 두 번째 패킷은 송신처가 웹 서버를 나타내지만 TCP 컨트롤 비트의 조건이 두 번째 행과 합치되지 않으므로, 세 번째 행의 조건에 합치되어 패킷 필터링을 통과한다.
- 그 후의 패킷도 첫 번째 행이나 세 번째 행 중 어느 하나에 합치된다.
- 결국 인터넷측에서 웹 서버에 접속할 때 흐르는 패킷은 전부 패킷 필터링을 통과한다.

컨트롤 비트

- 실제로는 통과시키는 것과 차단하는 것을 완전히 선별할 수 없는 경우도 있는데, DNS 서버에 대한 접속이 대표적인 예이다.
- DNS 서버에 조회하는 동작은 UDP를 사용하는데, UDP는 TCP와 달리 접속 단계의 동작이 없으므로 TCP처럼 컨트롤 비트에 의해 접속 방향을 판별할 수 없다.
- 그러므로 사내에서 인터넷의 DNS 서버에 접속하는 것을 허가하고, 인터넷에서 사내의 DNS 서버에 접근하는 패킷은 차단한다는 조건을 설정할 수 없다.
- 이 성질은 DNS 뿐만 아니라 UDP를 사용하는 애플리케이션에 공통이다.
- 이와 같은 경우에는 어느 정도 위험을 각오한 상태에서 애플리케이션의 패킷을 전부 통과시키거나 불편을 감수하고 애플리케이션을 전면적으로 차단하는 방법을 선택해야 한다.

사내 LAN에서 공개 서버용 LAN으로 조건을 설정

- 패킷 필터링의 그림과 같은 구성의 경우 인터넷과 공개 서버용 LAN을 왕래하는 패킷의 조건을 설정할 뿐만 아니라 사내 LAN과 인터넷 또는 사내 LAN과 공개 서버용 LAN을 왕래한 패킷의 조건도 설정해야 한다.
- 이때 조건이 서로 악영향을 끼치지 않도록 주의해야 한다.
- 예를 들어 사내 LAN과 공개 서버용 LAN 사이를 자유로이 왕래할 수 있도록 수신처 IP 주소가 공개 서버용 LAN과 일치하는 패킷을 전부 통과시켰다고 가정해 보자.
- 그런 다음 깜빡 잊고 송신처 IP 주소를 조건으로 설정하지 않으면 인터넷측에서 흘러온 패킷이 무조건 공개 서버용 LAN에 유입된다.
- 이렇게 되면 공개 서버용 LAN에 설치한 서버 전부가 위험한 상태에 빠지므로 이런 사태가 되지 않도록 신중하게 조건을 설정해야 한다.

2. 방화벽의 원리와 동작

밖에서 사내 LAN으로 액세스할 수 없다

- 패킷 필터링형 방화벽은 패킷을 통과시킬지 차단시킬지를 판단할 뿐만 아니라 주소 변환의 기능도 가지고 있으므로 설정도 필요하다.
- 구체적으로는 패킷 필터링과 마찬가지로 패킷의 시점과 종점을 조건으로 지정한 후 주소 변환이 필요한 경우에는 주소 변환을 하고, 변환이 필요하지 않은 경우에는 변환하지 않도록 설정한다.
- 프라이빗 주소와 글로벌 주소의 대응이나 포트 번호의 대응은 자동으로 할 수 있으므로 주소 변환을 해야 할지 설정한다.
- 주소 변환의 구조를 생각했지만, 주소 변환을 이용하면 당연히 인터넷측에서 사내 LAN에는 접근할 수 없게 된다.
- 따라서 사내 LAN에 대한 접근을 금지하도록 패킷 필터링의 조건을 설정할 필요가 없다.

2. 방화벽의 원리와 동작

방화벽 통과

- 방화벽에는 여러 가지 조건이 설정되어 있다.
- 여기에 패킷이 도착하면 조건에 해당하는지 판정하고, 통과시킬지와 차단할지를 결정한다.
- 이렇게 판정한 후 차단하는 대상이 되면 패킷을 버리고, 버린 기록을 남긴다.
- 버린 패킷 중에는 부정 침입의 흔적을 나타내는 것이 있으므로 이것을 분석하여 침입자의 수법을 밝히거나 향후 부정 침입 대책에 도움이 될 수 있기 때문이다.
- 통과시킨다는 판정을 내린 경우에는 패킷을 중계하는데, 이 중계 동작은 라우터의 동작과 같다.
- 일단 통과시킨다고 결정되면 그 이상의 특별한 구조는 없다.
- 복잡한 조건 설정이나 버린 패킷의 기록이 필요하지 않으면 패킷 필터링 기능을 가진 라우터를 방화벽으로 사용할 수도 있다.

2. 방화벽의 원리와 동작

방화벽으로 막을 수 없는 공격

- 방화벽은 패킷 흐름의 시점과 종점을 보고 통과시킬지, 차단할지 판단하는데. 시점과 종점만 보고 위험한 패킷을 전부 판단할 수 있는 것은 아니다.
- 예를 들어 웹 서버에 좋지 않은 상태가 있어서 특수한 데이터를 포함한 패킷을 받으면 웹 서버가 다운된다는 상황을 생각해 보자.
- 방화벽은 시점과 종점만 조사하므로 패킷 중에 특수한 데이터가 포함되어 있어도 이것에 신경쓰지 않고 패킷을 통과시켜 버린다.
- 그러면 이 패킷이 웹 서버에 도착하여 웹 서버는 다운된다.
- 이 예에서 알 수 있듯이 패킷의 내용을 조사하지 않으면 위험한지 판단할 수 없어서 방화벽의 구조는 이러한 상황에 대처할 수 없게 된다.
- 이러한 상황에 대해 두 가지의 대처법이 있다.
- 이 문제의 원인은 웹 서버 소프트웨어의 버그에 있으므로 버그를 고쳐서 다운되지 않도록 하는 것이 한 가지 대처법이다.
- 또 한 가지 방법은 패킷의 내용을 조사하여 위험한 데이터가 포함되어 있는 경우에 패킷을 차단하도록 장치나 소프트웨어를 방화벽과는 별도로 준비하는 방법이다.

2. 방화벽의 원리와 동작

방화벽으로 막을 수 없는 공격

- 이 방법은 패킷의 내용까지 조사하므로 완벽하다고 단순하게 생각하면 안 된다.
- 패킷의 내용이 위험한지는 웹 서버에 버그가 있는 지의 여부로 결정된다.
- 그러므로 잠재적인 버그가 숨어있는데도 아직 버그가 발견되지 않은 경우에는 패킷이 위험하다고 판단할 수 없기 때문에 패킷을 차단할 수 없다.
- 이 점은 발견된 버그를 고치는 방법과 크게 다르지 않다.
- 단 여러 대의 서버가 있는 경우에는 새로운 버전으로 바꾸기를 미루거나 잊어버리기 쉬우므로 패킷의 내용을 검사하는 방법은 효과가 있다.

3. 서버의 부하 분산

처리 능력이 부족하면 복수 서버로 부하 분산

- 서버에 접근하는 트래픽이 증가할 때는 서버로 통하는 회선을 빠르게 하는 방법이 효과적이지만, 회선을 빠르게 하는 것만으로 부족한 경우도 있다.
- 고속화한 회선에서 흘러오는 대량의 패킷에 서버의 처리 능력이 따라잡지 못할 수도 있기 때문이다.
- 이때 가장 먼저 떠오르는 방법은 서버 머신을 고성능 기종으로 교체하는 것이다.
- 하지만, 다수의 사용자가 집중적으로 접속하면 아무리 고성능의 기종을 사용해도 한 대로는 따라잡지 못할 수 있다.
- 이때는 복수의 서버를 사용하여 처리를 분담하는 방법으로 서버 한 대당 처리량을 줄이는 것이 효과적이다.
- 이러한 처리 방법을 통틀어 분산 처리라고 하는데 처리를 분담하는 방식은 여러 가지이다.
- 가장 간단한 방법은 단순히 여러 대의 웹 서버를 설치하고 한 대가 담당하는 사용자 수를 줄이는 방법이다.
- 이 방법을 취할 경우 클라이언트가 보내는 요청을 웹 서버에 분배하는 구조가 필요하다.
- 구체적인 방법은 몇 가지가 있는데 DNS 서버에서 분배하는 방법이 가장 간단하다.

3. 서버의 부하 분산

처리 능력이 부족하면 복수 서버로 부하 분산

- 서버에 액세스할 때 DNS 서버에 조회하여 IP 주소를 조사하는데, DNS 서버에 같은 이름으로 여러 대의 웹 서버를 등록해 놓으면 DNS 서버는 조회가 있을 때마다 차례대로 IP 주소를 되돌려준다.
- 예를 들어 www.naver.com 이라는 서버명에 대해 다음과 같은 3개의 IP 주소를 대응시킨다고 가정해 보자.

```
192.0.2.60
192.0.2.70
192.0.2.80
```

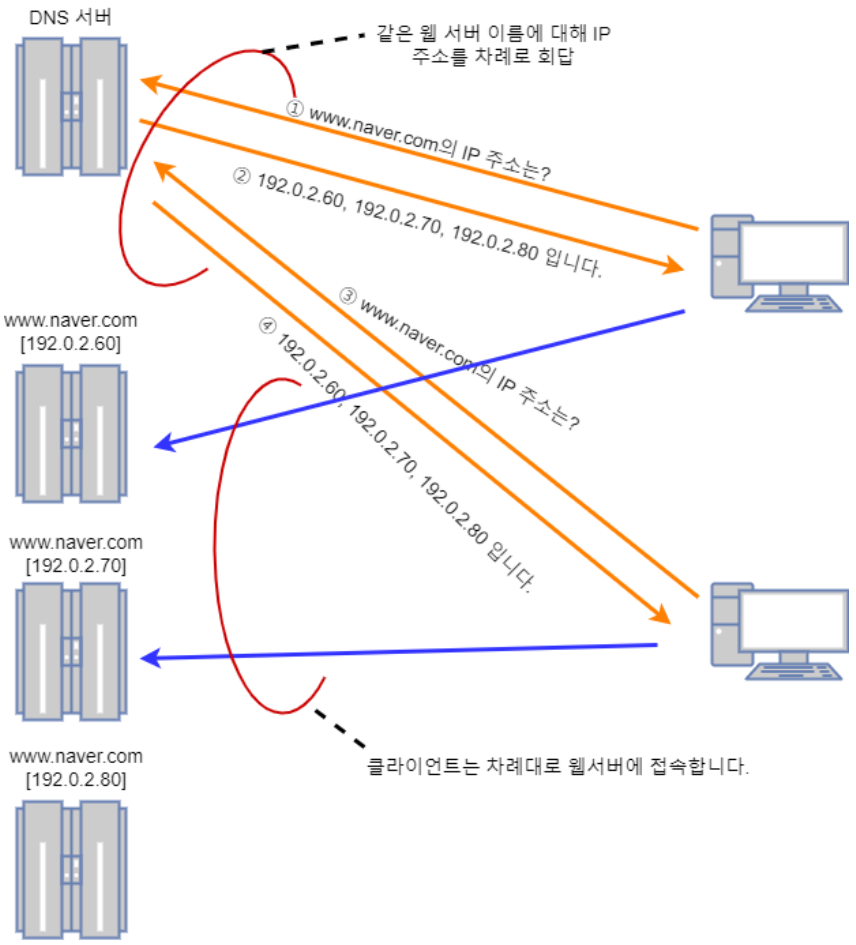
- 주소 조회를 순서를 바꿔가며 순환하는데 이 방법을 라운드 로빈이라고 한다.
- 이렇게 해서 복수의 서버에 균등하게 액세스를 분산시킬 수 있다.
- 이 방법에는 결점이 있다.
- 웹 서버가 많으면 이 중에서 고장나는 것도 있다.
- 이때 고장난 웹 서버를 피해서 IP 주소를 회답하면 좋지만, 보통의 DNS 서버는 웹 서버가 동작하지 않는지 확인하지 못하므로 웹 서버가 정지해도 상관하지 않고 IP 주소를 회답해 버린다.

3. 서버의 부하 분산

■ 처리 능력이 부족하면 복수 서버로 부하 분산

▪ 라운드 로빈

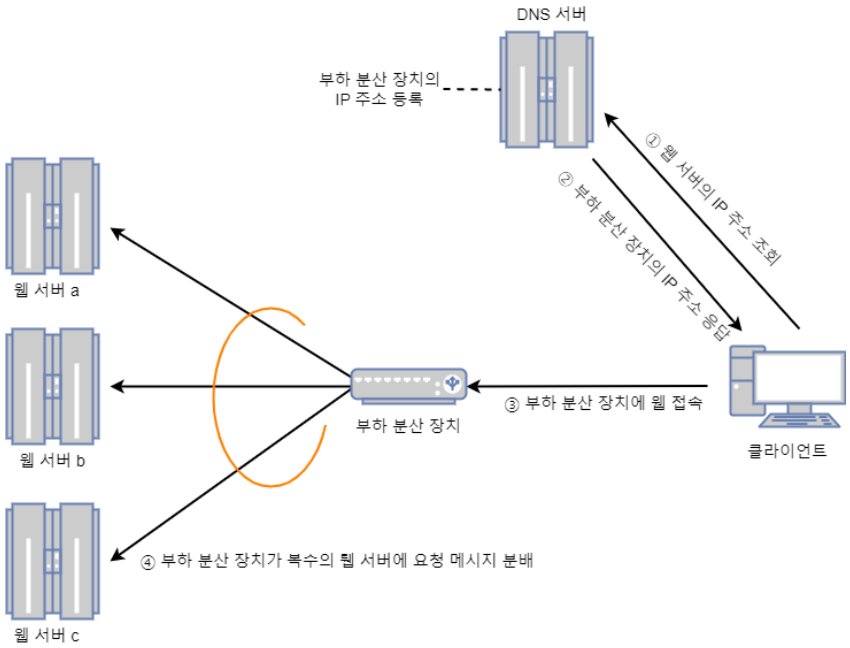
이름	클래스	타입	클라이언트에 회답하는 내용
www.naver.com	IN	A	192.0.2.60
www.naver.com	IN	A	192.0.2.70
www.naver.com	IN	A	192.0.2.80
…	…	…	…



3. 서버의 부하 분산

부하 분산 장치를 이용해 복수의 웹 서버로 분할

- 이러한 좋지 않은 상태를 피하기 위해 부하 분산 장치 또는 로드 밸런서 등으로 부르는 기기가 고안되었다.
- 부하 분산 장치를 사용할 때는 먼저 부하 분산 장치를 웹 서버 대신 DNS 서버에 등록한다.
- 그러면 클라이언트는 부하 분산 장치가 웹 서버라고 생각하여 여기에 요청 메시지를 보내야 할지 판단하고, 웹 서버에 요청 메시지를 전송한다(그림).
- 여기에서의 요점은 말할 것도 없이 어느 웹 서버에 요청 메시지를 전송해야 할지 판단하는 부분이다.



3. 서버의 부하 분산

부하 분산 장치를 이용해 복수의 웹 서버로 분할

- 판단 근거는 여러 가지가 있지만 대화가 복수의 페이지에 걸쳐있는지에 따라 판단 기준이 전혀 다르다.
- 복수 페이지에 걸쳐있지 않은 단순한 접속이라면 웹 서버의 부하 상태가 판단 근거가 될 것이다.
- 웹 서버와 정기적으로 정보를 교환하여 CPU나 메모리의 사용률 등을 수집하고, 이것을 바탕으로 어느 웹 서버의 부하가 낮은지 판단하거나 시험 패킷을 웹 서버에 보내 응답 시간으로 부하를 판단하는 방법이 일반적이다.
- 대화가 복수 페이지에 걸쳐있을 때는 웹 서버의 부하에 관계없이 이전의 요청과 같은 웹 서버에 전송해야 한다.
- 이렇게 하려면 먼저 대화가 복수 페이지에 걸쳐있는지 판단해야 하는 것이 요점이다.
- 하지만 HTTP는 stateless 프로토콜이다.
- 그러므로 웹 서버측에서 보면 HTTP의 대화는 1회씩 전혀 다른 것으로 보여서 받은 요청 메시지가 이전 요청에 계속되는 것인지, 아니면 이전 요청 메시지와는 관계가 없는 것인지를 판단할 수 없다.

3. 서버의 부하 분산

부하 분산 장치를 이용해 복수의 웹 서버로 분할

- 전후 관계를 알지 않아도 요청 메시지의 송신처 IP 주소가 같다면 일련의 것이라는 식으로 간단히 판단하면 되지만, 그럴 수도 없다.
- 프록시를 사용하면 요청 메시지의 송신처 IP 주소가 프록시의 IP 주소로 되어 실제로 요청 메시지를 보낸 클라이언트가 누구인지 모르게 되기 때문이다.
- 주소 변환을 이용하는 경우도 송신처 IP 주소가 주소 변환 장치의 IP 주소로 되므로 클라이언트를 판별할 수 없다.
- 전후 관계를 판단하기 위해 여러 가지 방법이 고안되었다.
- 양식에 입력한 데이터를 보낼 때 그 안에 전후의 관련을 나타내는 정보를 부가하거나 HTTP의 사양을 확장하여 전후 관계를 판단하기 위한 정보를 HTTP 헤더필드에 부가하는 방법이다.
- 부하 분산 장치는 이러한 정보를 조사하여 일련의 동작이라면 이전과 같은 웹 서버에 요청 메시지를 전송하고, 그렇지 않으면 부하가 적은 웹 서버에 전송하도록 동작한다.

4. 캐시 서버를 이용한 서버의 부하 분산

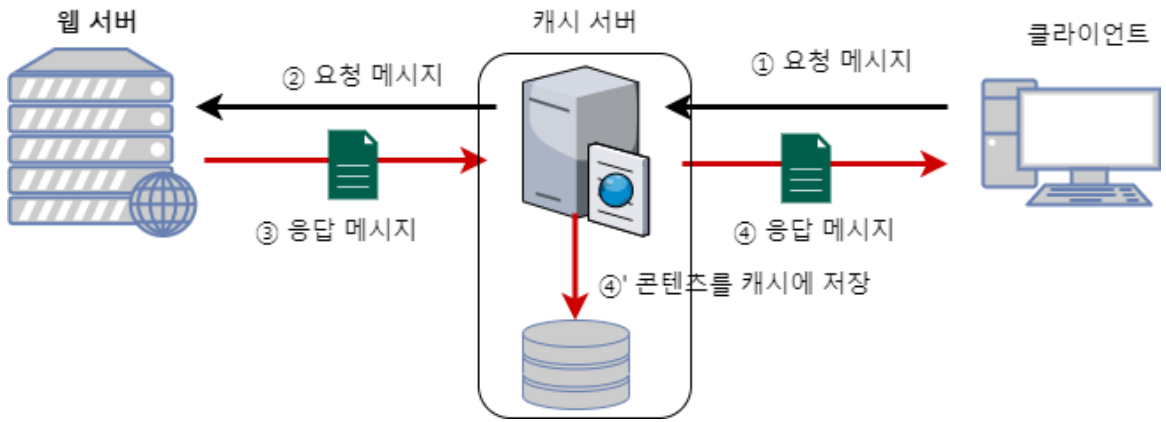
캐시 서버의 이용

- 여러 대의 웹 서버를 설치하는 즉 같은 기능을 가진 여러 대의 서버를 설치하는 것이 아니라 다른 방법으로 부하 분산을 하는 방법도 있다.
- 이것은 데이터베이스 서버와 웹 서버 같은 역할에 따라 서버를 나누는 방법으로 이러한 역할별 분산 처리 방법 중의 하나가 캐시 서버를 사용하는 방법이다.
- 캐시 서버는 프록시라는 구조를 사용하여 데이터를 캐시에 저장하는 서버이다.
- 프록시는 웹 서버와 클라이언트 사이에 들어가서 웹 서버에 대한 액세스 동작을 중개하는 역할을 하는데 액세스 동작을 중개할 때 웹 서버에서 받은 데이터를 디스크에 저장해 두고 웹 서버를 대신하여 데이터를 클라이언트에 반송하는 기능을 가지고 있다.
- 이것을 캐시라고 부르며, 캐시 서버는 이 기능을 이용한다.
- 웹 서버는 URL을 점검하거나, 액세스 권한을 점검하거나, 페이지 안에 데이터를 내장하는 등의 처리를 내부에서 실행하기 위해 페이지의 데이터를 클라이언트에 송신할 때 다소 시간이 걸린다.
- 한편 캐시 서버쪽은 웹 서버에서 받아 보존해 둔 데이터를 읽어서 클라이언트에 송신만 하므로 웹 서버보다 빨리 데이터를 송신할 수 있다.

4. 캐시 서버를 이용한 서버의 부하 분산

캐시 서버는 갱신일로 콘텐츠를 관리

- 캐시 서버를 사용할 때는 부하 분산 장치와 마찬가지로 캐시 서버를 웹 서버 대신 DNS 서버에 등록한다.
- 그러면 사용자는 캐시 서버에 HTTP의 요청 메시지를 보낼 것이고(그림 ①), 캐시 서버가 메시지를 받는다.
- 이때의 수신 동작은 웹 서버의 수신 동작과 같다.
- 사용자가 보면 캐시 서버가 웹 서버로 보일 것이다.
- 그래서 요청 메시지를 받으면 캐시 서버는 요청 메시지의 내용을 조사하고 데이터가 자신의 캐시에 저장되었는지 조사한다.
- 저장되어 있지 않은 경우, 캐시 서버는 요청 메시지에 캐시 서버를 경유한 것을 나타내는 'Via' 라는 헤더 필드를 추가하여 웹 서버에 요청 메시지를 전송한다(그림 ②).



4. 캐시 서버를 이용한 서버의 부하 분산

캐시 서버는 갱신일로 콘텐츠를 관리

- 우선 어느 웹 서버에 요청 메시지를 전송해야 할지 판단해야 한다.
- 웹 서버가 한 대밖에 없으면 웹 서버의 도메인명이나 IP 주소를 캐시 서버에 설정해 두고 무조건 거기에 전송하는 방법을 취한다.
- 그러나 한 대의 캐시로 여러 대의 서버의 데이터를 캐시에 저장하는 경우에는 이러한 단순한 방법으로는 곤란하다.
- 요청 메시지의 내용에 따라 전송 대상의 웹 서버를 판단하는 것 같은 방법이 필요하다.
- 이 방법은 몇 가지가 있는데, 대표적인 것은 요청 메시지에 쓰여있는 디렉토리를 보고 판단하는 방법이다.
- 이 방법을 사용하는 경우에는 먼저 캐시 서버에 다음과 같이 전송 대상을 설정한다.

URI가 /dir1/이라는 디렉토리였다면 www1.naver.com에 전송한다.
URI가 /dir2/라는 디렉토리였다면 www2.naver.com에 전송한다.

- 이 설정을 보고 전송 대상을 판단하여 요청 메시지를 전송한다.
- 이때 전송 대상의 웹 서버에 대해 캐시 서버가 클라이언트가 되어 요청 메시지를 보낸다.

4. 캐시 서버를 이용한 서버의 부하 분산

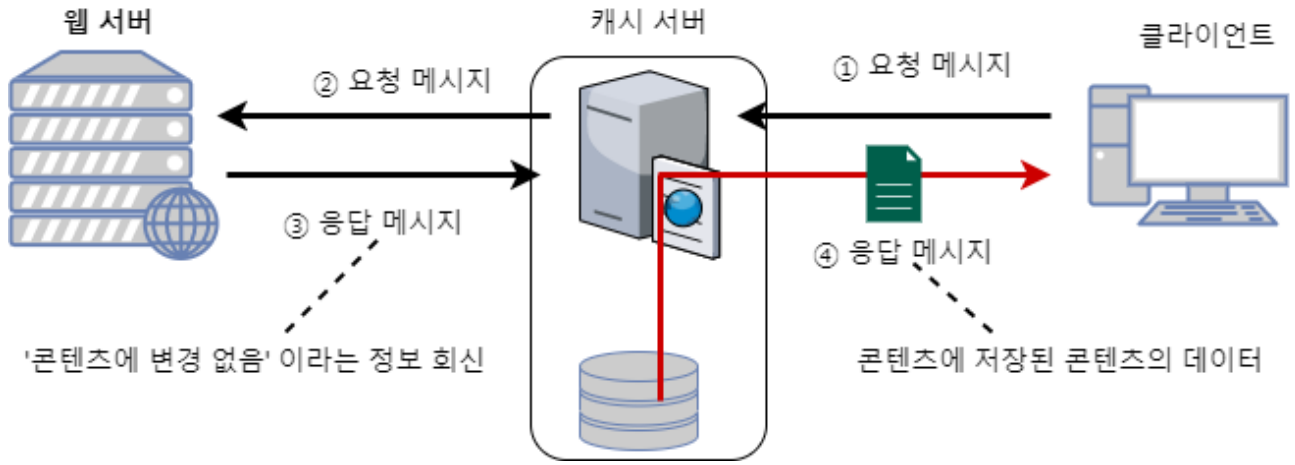
캐시 서버는 갱신일로 콘텐츠를 관리

- 웹 서버에서 보면 캐시 서버가 클라이언트에 보이는데, 웹 서버에서 캐시 서버에 응답 메시지가 돌아오므로 그것을 받는다(그림 ③).
- 캐시 서버는 응답 메시지에 캐시 서버를 경유한 것을 나타내는 'Via' 헤더 필드를 추가하며 클라이언트에 대해 웹 서버가 되어 응답 메시지를 전송한다(그림 ④).
- 그리고 응답 메시지를 캐시에 저장하고 저장한 일시를 기록한다(그림 ④').
- 이렇게 클라이언트와 웹 서버 사이를 중개하는 것이 프록시 구조이다.
- 그리고 중개할 때 페이지의 데이터를 저장하면 캐시 서버에 데이터가 축적되고, 사용자가 접근한 페이지가 캐시 서버에 저장되어 있는 경우가 증가한다.

4. 캐시 서버를 이용한 서버의 부하 분산

캐시 서버는 갱신일로 콘텐츠를 관리

- 다음에는 데이터가 저장되어 있는 경우이다.



- 먼저 사용자로부터 도착한 요청 메시지를 받아 캐시에 저장되었는지 조사하는 곳은 앞의 설명과 같습니다(그림 ①).
- 그리고 웹 서버측에서 데이터가 변경되었는지 조사하기 위한 'If - Modified - Since' 라는 헤더 필드를 추가하여 웹 서버에 전송한다(그림 ②).
- 웹 서버는 'If - Modified -Since' 헤더 필드의 값과 페이지 데이터의 최종 갱신 일시를 비교하여 변경이 없으면 변경이 없는 것을 나타내는 응답 메시지를 반송한다(그림 ③).
- 이때 웹 서버는 데이터의 최종 갱신 일시를 조사하는 것으로 끝나므로 페이지 데이터를 반송하는 것보다 부담이 적어진다.

4. 캐시 서버를 이용한 서버의 부하 분산

- 캐시 서버는 갱신일로 콘텐츠를 관리
  - 또한 반송 응답 메시지도 짧으므로 부담도 적어진다.
  - 이후 응답 메시지는 캐시 서버에 도착할 것이다.
  - 이렇게 해서 캐시에 저장한 데이터가 최신 데이터와 같은 것을 알 수 있으므로 캐시 서버는 캐시에서 데이터를 추출하여 사용자에게 보낸다(그림 ④).
  - 이 응답 메시지의 내용은 캐시에 데이터가 없었던 경우와 같다.
  - 웹 서버측에서 데이터가 변경된 경우에는 캐시에 데이터가 저장되어 있지 않은 경우와 같다.
  - 웹 서버는 최신 데이터를 반송하므로(그림 ③), 'Via' 헤더를 부가하여 사용자에게 전송하고 데이터를 캐시에 저장한다.

4. 캐시 서버를 이용한 서버의 부하 분산

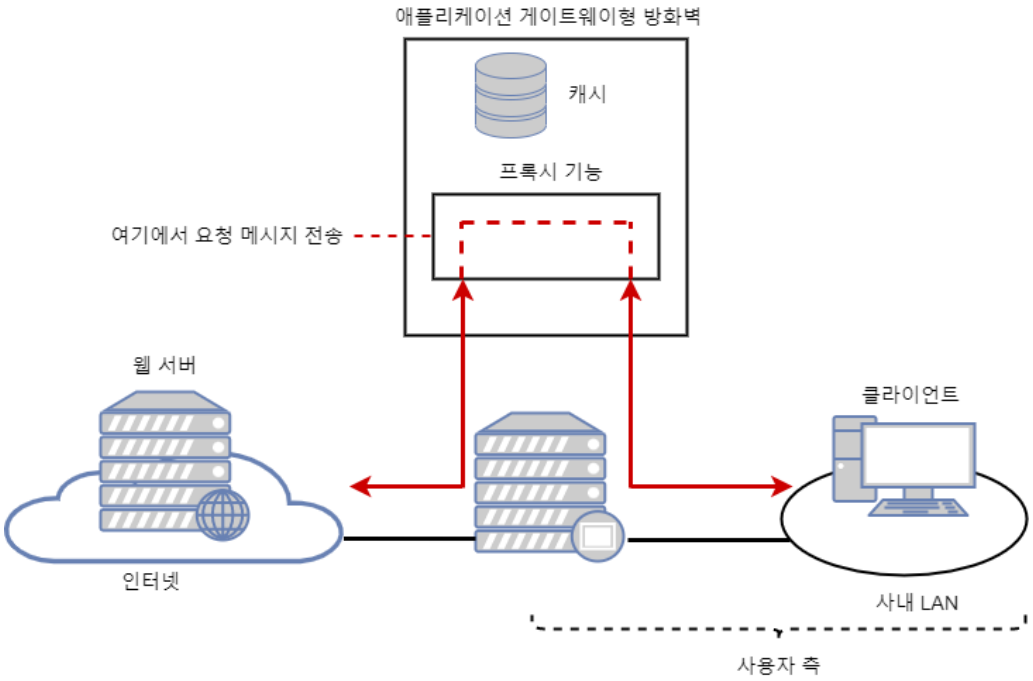
프록시의 원점, 포워드 프록시

- 지금까지 설명한 것은 프록시라는 구조를 웹 서버측에 두고 캐시 기능을 이용하는 것이지만, 클라이언트측에 캐시 서버를 두는 방법도 있다.
- 사실 캐시 서버에서 이용하는 프록시라는 구조는 원래 클라이언트측에 두는 방법에서 시작되었다.
- 이 유형이 프록시의 원형으로, 포워드 프록시라고 한다.
- 포워드 프록시가 처음 등장했을 때 캐시를 이용하는 것이 목적이었다.
- 이것은 서버측에 설치하는 캐시 서버와 같지만, 당시의 포워드 프록시는 방화벽을 실현한다는 중요한 목적이 한 가지 더 있었다.
- 방화벽의 이용 목적은 인터넷에서의 부정 침입을 막는 것이지만, 이 목적을 달성하는 가장 현실적인 방법은 인터넷과 사내의 패킷 왕래를 전부 정지시키는 것이다.
- 그러나 패킷을 전부 정지시키면 인터넷에 대한 접속도 정지된다.
- 이렇게 하면 도움이 되지 않으므로 필요한 것을 통과시키는 방법을 생각하기 위해 '프록시'라는 구조를 고안한 것이다.

4. 캐시 서버를 이용한 서버의 부하 분산

프록시의 원점, 포워드 프록시

- 그림과 같이 프록시에서 요청 메시지를 일단 받아서 인터넷을 향해 전송하면 필요한 것을 통과시킬 수 있다는 개념이다.
- 이때 프록시의 캐시를 이용하면 더 효과적이다.
- 이전에 접속한 페이지의 경우 사내 LAN에서 프록시에 접속하기만 하면 데이터를 손에 넣을 수 있으므로 저속 회선에서 인터넷에 접속하는 것보다 매우 빨라질 것이다.



4. 캐시 서버를 이용한 서버의 부하 분산

프록시의 원점, 포워드 프록시

- 프록시는 요청 메시지의 내용을 조사한 후 전송하므로 요청 메시지의 내용에 따라 접근이 가능한지 판단할 수 있다.
- 즉 위험한 사이트나 작업과 관계없는 사이트에 대한 접근은 금지한다는 접근 제한을 마련할 수 있다.
- 패킷 필터링형 방화벽이라면 판단 근거로 IP 주소나 포트 번호라는 정보만 사용하므로 이렇게까지 자세히 조건을 설정하는 것은 불가능하다.
- 포워드 프록시를 사용할 경우에는 보통 브라우저의 설정 화면에 준비되어 있는 프록시 서버라는 항목에 포워드 프록시의 IP 주소를 설정한다.
- 그러면 브라우저의 요청 메시지 송신 동작이 조금 달라진다.
- 포워드 프록시를 설정하지 않으면 브라우저는 URL 입력 상자에 입력된 http:// .... 라는 문자열에서 접속 대상의 웹 서버를 계산하고 여기에 요청 메시지를 보낸다.
- 하지만 포워드 프록시를 설정하면 URL의 내용에 상관없이 요청 메시지를 전부 포워드 프록시에 보낸다.

4. 캐시 서버를 이용한 서버의 부하 분산

프록시의 원점, 포워드 프록시

- 그리고 요청 메시지의 내용도 조금 다른 형태가 된다.
- 포워드 프록시를 설정하지 않는 경우에는 URL에서 웹 서버의 이름을 제외하고 파일이나 프로그램의 경로명의 일부를 추출하여 이것을 요청 메시지의 URI 부분에 기록한다.
- 하지만 포워드 프록시를 설정하면 http:// ..... 라는 URL을 그대로 요청 메시지의 URL에 기록한다.
- 포워드 프록시는 메시지를 전송하는 동작도 서버측에 두는 캐시 서버와 전송 대상의 웹 서버를 판단하는 부분이 약간 다르다.
- 포워드 프록시를 사용할 경우에는 URI의 부분에 http:// .... 라는 URL이 그대로 쓰여 있으므로 URL이 전송 대상이 된다.
- 그렇기 때문에 서버측에 두는 캐시 서버와 같이 전송 대상의 웹 서버를 사전에 설정해 둘 필요는 없고 모든 웹 서버에서 전송할 수 있다.
- 서버측에 두는 캐시 서버라면 설정해 둔 웹 서버에만 전송할 수 있으므로 상당히 다르다.

4. 캐시 서버를 이용한 서버의 부하 분산

포워드 프록시를 개량한 리버스 프록시

- 포워드 프록시를 사용할 경우에는 브라우저에 대한 설정이 꼭 필요한데, 이것이 포워드 프록시의 특징이다.
- 그런데 이 방법은 브라우저의 설정이 번거롭고 잘못 설정할 경우에는 브라우저가 제대로 작동하지 않는 장애의 원인이 되기도 한다.
- 브라우저에 설정이 필요하다는 점은 이런 수고나 장애의 원인 뿐만 아니라 다른 제약 사항이 되기도 한다.
- 인터넷에 공개하는 웹 서버는 누가 액세스하는지 알 수 없고, 브라우저에 프록시를 설정할 수 없기 때문에 웹 서버의 바로 앞에 프록시를 두는 방법을 선택하지 않는다.
- 이에 따라 브라우저에 프록시를 설정하지 않아도 사용할 수 있도록 개량되었다.
- 즉 요청 메시지의 URI에 쓰여있는 디렉토리명과 전송 대상의 웹 서버를 대응시켜서 URI 부분에 http:// ..... 라고 쓰여있지 않은 보통의 요청 메시지를 전송할 수 있도록 했다.
- 이것이 서버측에 설치하는 캐시 서버에 채택하고 있는 방식으로 리버스 프록시라고 부른다.

4. 캐시 서버를 이용한 서버의 부하 분산

트랜스페어런트 프록시

- 캐시 서버에서 전송 대상을 판단하는 방법, 즉 요청 메시지에서 패킷의 헤더를 조사하는 방법이 있다.
- 패킷의 맨 앞에 있는 IP 헤더에는 수신처 IP 주소가 기록되어 있으므로 이것을 조사하면 액세스 대상 웹 서버가 어디에 있는지 알 수 있는데, 이 방법을 트랜스페어런트 프록시라고 부른다.
- 이 방법이라면 보통의 요청 메시지를 전송할 수 있으므로 포워드 프록시처럼 브라우저에 설정할 필요가 없다.
- 또한 전송 대상을 캐시 서버에 설정할 필요도 없고, 어느 웹 서버에서나 전송할 수 있다.
- 트랜스페어런트 프록시는 포워드 프록시와 리버스 프록시의 좋은 점만 모은 형태의 편리한 구조이지만 트랜스페어런트 프록시에 요청 메시지를 건네주는 방법을 주의해야 한다.
- 트랜스페어런트 프록시는 브라우저에 설정하지 않으므로 브라우저는 웹 서버에 요청 메시지를 보낸다.
- 리버스 프록시와 같이 DNS 서버에 등록하는 방법이라면 이 요청 메시지가 프록시에 도착하지만 트랜스페어런트 프록시는 DNS 서버에 등록하는 것도 없다.

4. 캐시 서버를 이용한 서버의 부하 분산

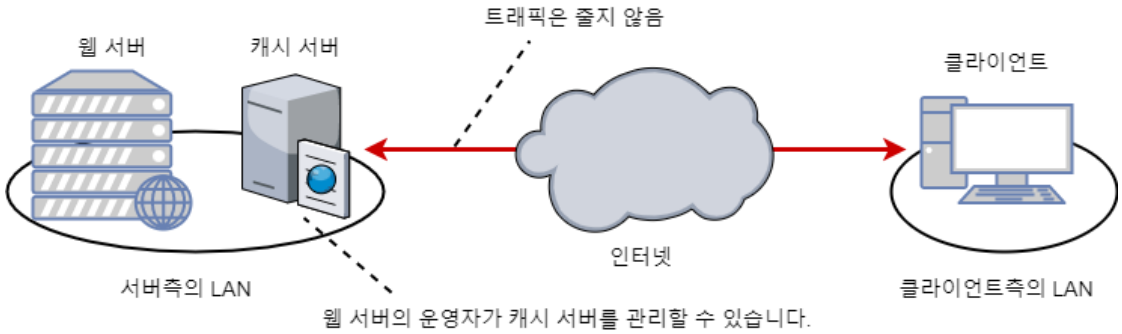
트랜스페어런트 프록시

- DNS 서버에 등록하면 트랜스페어런트 프록시 자체가 액세스 대상이 되어 수신처 IP 주소로 전송 대상의 웹 서버를 판단한다는 중요한 구조를 이용할 수 없게 된다.
- 이대로라면 요청 메시지는 브라우저에서 웹 서버로 흘러갈 뿐 트랜스페어런트 프록시에는 도착하지 않는다.
- 이것을 해결하기 위해 브라우저에서 웹 서버로 요청 메시지가 흘러가는 길에 트랜스페어런트 프록시를 설치한다.
- 그리고 메시지가 트랜스페어런트 프록시를 통과할 때 그것을 가로챈다.
- 편법이라는 느낌도 있지만 이렇게 해서 요청 메시지가 트랜스페어런트 프록시에 도착하고, 웹 서버에 전송할 수 있다.
- 또한 요청 메시지가 흐르는 길이 많으면 여기에 전부 트랜스페어런트 프록시를 설치해야 하므로 길이 한 개로 수렴하는 형태로 네트워크를 만들고, 수렴되는 곳에 트랜스페어런트 프록시를 설치하는 것이 보통이다.
- 인터넷 접속 회선 부분이 이러한 형태로 되어 있으므로 접속 회선 부분에 설치해도 된다.
- 트랜스페어런트 프록시를 사용하면 사용자가 프록시의 존재를 알아차릴 필요가 거의 없다.
- 따라서 HTTP의 메시지를 전송한다는 구조에 대한 관심이 적어지고 캐시를 이용한다는 측면에서 비중이 높아지고 있다.

5. 콘텐츠 배포 서비스

콘텐츠 배포 서비스를 이용한 부하 분산

- 캐시 서버는 서버측에 두는 경우와 클라이언트측에 두는 경우가 이용 효과면에서 차이가 난다.
- 그림의 (a)에서 알 수 있듯이 서버측에 캐시 서버를 두는 방법은 웹 서버의 부하를 경감하는 효과는 있지만, 인터넷을 흐르는 트래픽을 억제하는 효과는 없다.
- 이 점에서는 클라이언트측에 캐시 서버를 두는쪽이 낫다(그림 (b)).
- 인터넷 안에는 혼잡한 곳이 있을 수 있어서 그곳을 통과하려면 시간이 걸린다.
- 클라이언트측에 캐시 서버가 있으면 이러한 혼잡에 휘말려드는 일이 없으므로 패킷의 흐름이 안정된다.
- 큰 회상이나 영상 같은 대용량 데이터를 포함하는 콘텐츠라면 효과가 크다.

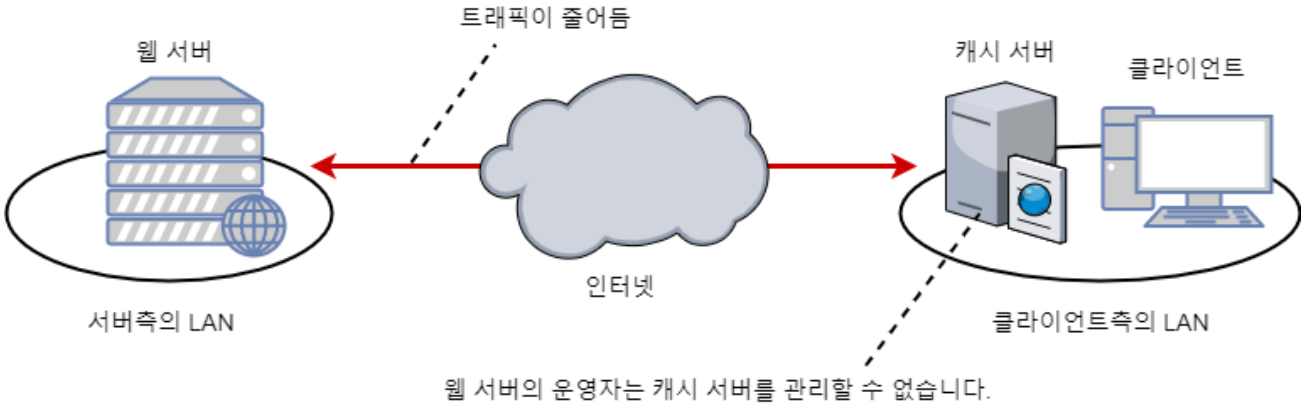


(a) 웹 서버의 직전에 캐시 서버를 두는 경우

5. 콘텐츠 배포 서비스

콘텐츠 배포 서비스를 이용한 부하 분산

- 클라이언트측에 두는 캐시 서버는 클라이언트측의 네트워크를 운영 관리하는 사람이 소유하므로 웹 서버 운영자가 제어할 수 없다.
- 예를 들어 웹 서버 운영자가 최근 콘텐츠의 용량이 커져서 캐시 서버의 용량을 높인다고 가정해 보자.
- 서버측의 캐시 서버라면 자신이 소유하고 있으므로 디스크를 증설해서 용량을 늘리면 되지만, 클라이언트측의 캐시 서버는 그럴 수가 없다.
- 무엇보다도 클라이언트측에 캐시 서버가 있다고 단정할 수 없다.

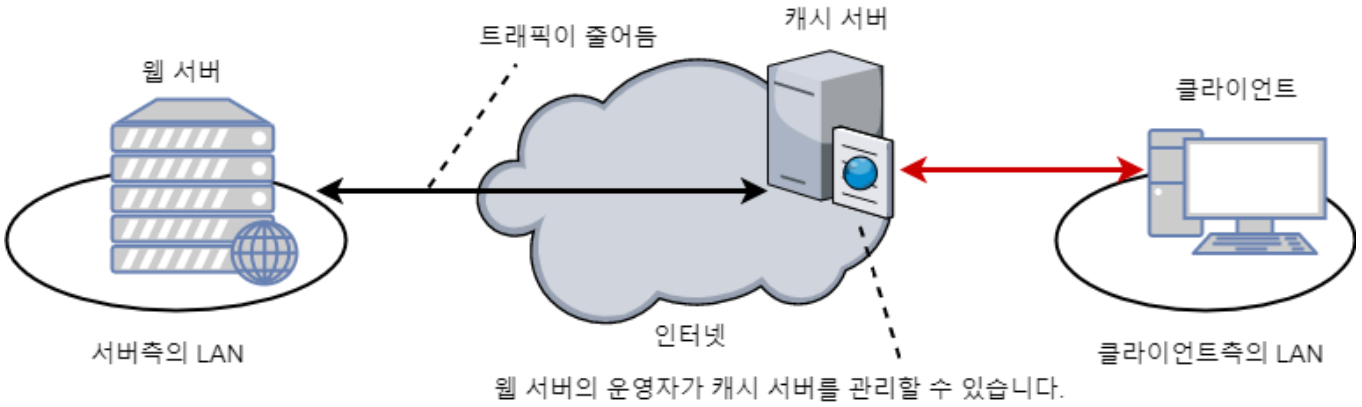


(b) 클라이언트측에 캐시 서버를 두는 경우

5. 콘텐츠 배포 서비스

콘텐츠 배포 서비스를 이용한 부하 분산

- 캐시 서버는 놓는 장소에 따라 장점과 단점이 있지만 양쪽의 좋은 점을 취한 방법도 있다 .
- 이것은 그림의 (c)와 같이 프로바이더와 계약하여 웹 서버 운영자가 제어할 수 있는 캐시 서버를 클라이언트측의 프로바이더에 두는 방법이다.
- 이 방법이라면 사용자로부터 가까운 장소에 캐시 서버를 설치하면서 이것을 웹 서버 운영자가 제어할 수 있지만 이 방법에도 문제가 있다.
- 인터넷에 공개하는 서버는 인터넷의 어디에서 접속하는지 알 수 없다.



(c) 인터넷 주위에 캐시 서버를 두는 경우

5. 콘텐츠 배포 서비스

콘텐츠 배포 서비스를 이용한 부하 분산

- 따라서 이 방법을 전면적으로 실현하려면 프로바이더의 POP 전부에 캐시 서버를 설치해야 하지만, 수가 너무 많으므로 비현실적이다.
- 이 문제를 해결하기 위해 먼저 중요한 프로바이더에 중점을 두면 캐시 서버의 수를 줄일 수 있다.
- 이렇게 하면 사용자에 따라서는 캐시 서버에 도착하기까지 먼 길을 거쳐야 한다.
- 그래도 웹 서버에 직접 접속하는 것보다는 여정을 단축할 수 있어서 나름대로 효과적이다.
- 이렇게 해서 현실성을 높일 수 있지만 또 한 가지 문제가 있다.
- 아무리 수를 줄여도 웹 서버 운영자가 스스로 프로바이더와 계약하여 캐시 서버를 설치하는 것은 비용면에서나 노력면에서 보통 일이 아닌데, 이것도 해결하는 방법이 있다.
- 즉 캐시 서버를 설치하고 이것을 웹 서버 운영자에게 대출하는 서비스를 제공하는 사업자가 등장했는데, 이런 종류의 서비스를 콘텐츠 배포 서비스(CDS; Content Delivery Service)라고 한다.

5. 콘텐츠 배포 서비스

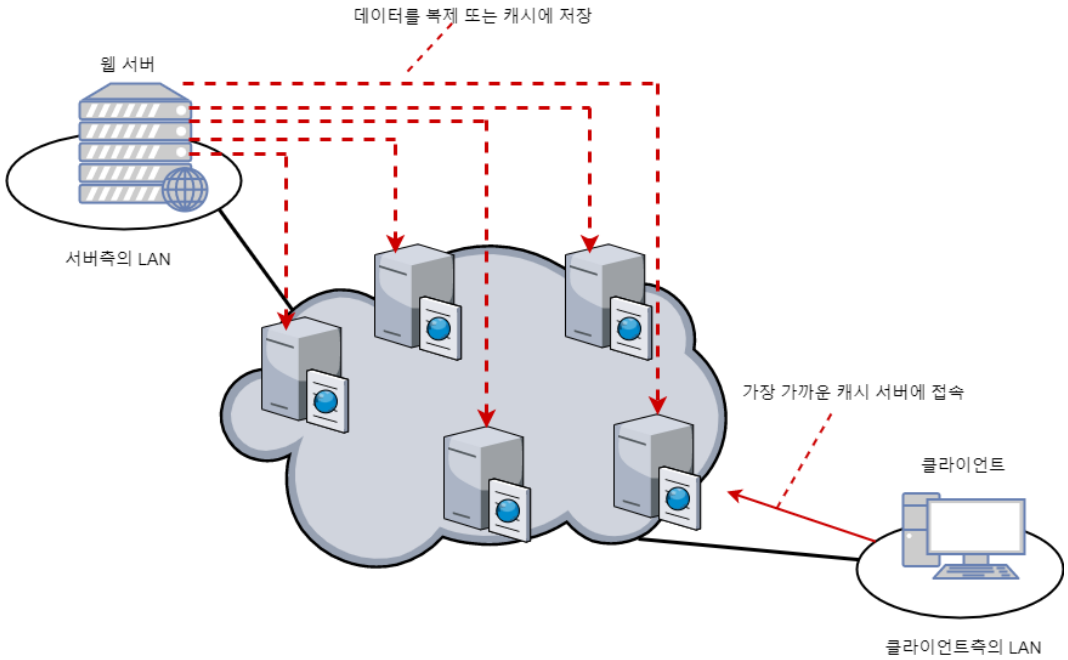
콘텐츠 배포 서비스를 이용한 부하 분산

- 이 서비스를 제공하는 사업자 CDSP(Content Delivery Service Provider)는 중요한 프로바이더와 계약하고 그곳에 다수의 캐시 서버를 설치하다.
- 한편 CDSP는 웹 서버 운영자와도 계약하여 웹 서버와 CDSP의 캐시 서버를 연대시킨다.
- 이것에 대한 구체적인 방법은 뒤에서 설명하지만 웹 서버와 캐시 서버를 잘 연대시키면 클라이언트가 웹 서버에 접속할 때 CDSP의 캐시 서버에 접속한다.
- 캐시 서버는 다수의 웹 서버의 데이터를 캐시에 저장할 수 있으므로 CDSP가 설치한 캐시 서버를 다수의 웹 서버 운영자가 공동으로 이용할 수도 있다.
- 그러면 웹 서버 운영자의 한 회사당 비용을 절감할 수 있는데, 이렇게 해서 웹 서버 운영자의 비용 부담이 줄어든다.
- 또한 프로바이더와의 계약은 CDSP가 한 번에 인수하므로 노력면에서도 웹 서버 운영자에 부담이 되지 않는다.

5. 콘텐츠 배포 서비스

가장 가까운 캐시 서버의 관점

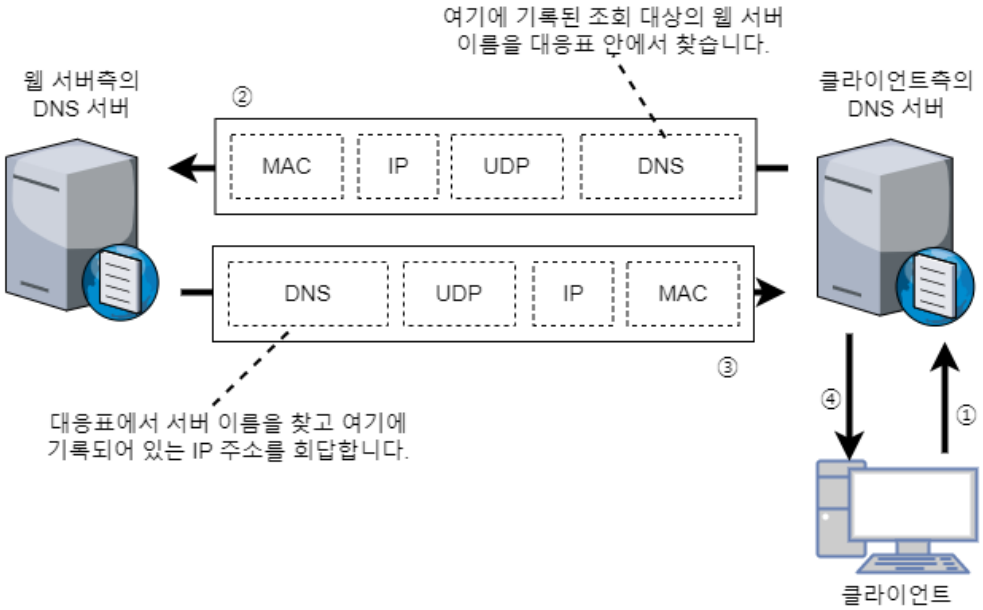
- 콘텐츠 배포 서비스를 사용하는 경우 그림과 같이 인터넷 전체에 설치된 다수의 캐시 서버를 이용한다.
- 이러한 상황에서는 다수가 있는 캐시 서버 중에서 가장 가까운 캐시 서버를 찾아내고 클라이언트가 여기에 액세스하도록 중재하는 구조가 필요하다.
- 포워드 프록시를 사용할 때와 같이 브라우저에 프록시 서버를 설정하면 좋지만, 모든 사용자가 따르게 하는 것은 현실적으로 불가능하다.
- 사용자가 아무리 따라주어도 요청 메시지가 캐시 서버에 도착하는 것 같은 중간 과정이 필요하다.



5. 콘텐츠 배포 서비스

가장 가까운 캐시 서버의 관점

- 최초의 방법은 복수 서버를 설치하여 부하를 분산시킬 때 DNS 서버에서 액세스를 분배하는 것과 비슷한 방법이다.
- 즉 DNS 서버가 웹 서버의 IP 주소를 회답할 때 가장 가까운 캐시 서버의 IP 주소를 회답하도록 DNS 서버를 세밀하게 설정하는 방법이다.
- 이 방법을 설명하기 전에 보통의 DNS 서버의 동작을 복습해 보자.
- DNS 서버는 인터넷에 다수 배치되어 있고 이것들이 연대하여 조회와 회답을 한다.
- 이 동작은 클라이언트가 조회 메시지를 보내는 곳부터 시작된다(그림①).



5. 콘텐츠 배포 서비스

가장 가까운 캐시 서버의 관점

- 그러면 클라이언트측의 DNS 서버는 웹 서버의 이름의 계층 구조를 조사하여 이름이 등록되어 있는 DNS 서버 즉 웹 서버측에 있는 DNS 서버를 찾아내고 거기에 조회 메시지를 보낸다(그림 ②).
- 웹 서버측의 DNS 서버는 조회 메시지를 받고 이름에 대응하는 IP 주소를 조사하여 회답한다.
- 웹 서버측의 DNS 서버에는 관리자가 등록한 서버명과 IP 주소를 대응시킨 대응표가 있을 것이므로 대응표에서 서버명을 찾고, 이것에 대응하는 IP 주소를 회답한다(그림 ③).
- 그러면 회답이 클라이언트측의 DNS 서버에 도착하고, 여기에서 클라이언트측에 회답이 되돌려진다(그림 ④).
- 여기까지가 보통의 DNS 서버의 동작이다.
- 이것을 그대로 사용하면 클라이언트측과 캐시 서버의 위치 관계를 전혀 고려하지 않고 라운드 로빈을 이용해 차례대로 IP 주소를 회답하기만 하므로 먼 위치에 있는 캐시 서버의 IP 주소를 되돌려줄지도 모른다.

5. 콘텐츠 배포 서비스

가장 가까운 캐시 서버의 관점

- 그래서 가장 가까운 캐시 서버에 접속할 경우에는 라운드 로빈이 아니라 클라이언트와 캐시 서버의 거리를 판단하여 클라이언트에 가장 가까운 캐시 서버의 IP 주소를 회답하도록 한다.
- 이때의 요점은 말할 것도 없이 클라이언트와 캐시 서버의 거리를 판단하는 방법이다.
- 이 방법은 다음과 같다.
- 먼저 준비 단계로 캐시 서버의 설치 장소에 있는 라우터에서 경로 정보를 모아둔다.
- 예를 들어 4대의 캐시 서버가 있으면 캐시 서버의 설치 장소에 있는 라우터는 4대가 되므로 각각의 라우터에서 경로표를 입수하여 4개의 경로표가 DNS 서버의 곁에 모인다.
- 이 경로표를 사용하여 DNS의 조회 메시지의 송신처, 즉 클라이언트측의 DNS 서버에 이르는 경로 정보를 조사한다.
- 이것을 모든 라우터에 대해 조사하고 비교하면 어느 라우터가 클라이언트측의 DNS 서버에 가장 가까운지 알 수 있다.

5. 콘텐츠 배포 서비스

가장 가까운 캐시 서버의 관점

- 경로표를 입수한 라우터는 캐시 서버의 설치 장소에 있고, 클라이언트측의 DNS 서버도 클라이언트와 같은 장소에 있다고 가정해 보자.
- 이 과정을 통해 캐시 서버와 클라이언트의 거리를 알 수 있어서 어느 캐시 서버가 가장 가까운 위치에 있는지도 알 수 있다.
- 실제로 클라이언트측의 DNS 서버는 반드시 클라이언트와 같은 장소에 있는 것이 아니므로 정확한 거리를 측정하지는 못하지만, 웬만큼 정확하게 거리를 측정할 수 있다.

5. 콘텐츠 배포 서비스

리피터용 서버로 액세스 대상을 분배

- 가장 가까운 캐시 서버에 액세스하는 방법은 한 가지가 더 있다.
- HTTP의 사양에는 다양한 헤더 필드가 정의되어 있고, 이 중에서 'Location' 이라는 헤더가 있다.
- 이것은 웹 서버의 데이터를 다른 서버로 옮기는 경우에 사용하는 것으로, '그 데이터는 이쪽의 서버에 있으므로 그쪽으로 다시 접속하세요.' 라는 의미이다.
- 이렇게 해서 다른 웹 서버에 액세스하도록 처리하는 것을 리다이렉트(redirect)라고 하며, 이것을 사용하여 액세스 대상을 가장 가까운 캐시 서버로 돌리는 것이 또 한 가지 방법이다.
- 리다이렉트에 의해 가장 가까운 캐시 서버를 클라이언트에 통지할 때는 리다이렉트용 서버를 웹 서버측의 DNS 서버에 등록한다.
- 그러면 클라이언트는 여기에 HTTP의 요청 메시지를 보낸다.
- 리다이렉트용 서버에는 DNS 서버와 같이 라우터에서 모은 경로 정보가 있으며, 여기에서 가장 가까운 캐시 서버를 찾는다.
- 그리고 캐시 서버를 나타내는 Location 헤더를 붙여 응답을 돌려보내면 클라이언트는 캐시 서버에 다시 액세스한다.

5. 콘텐츠 배포 서비스

리피터용 서버로 액세스 대상을 분배

- 이 방법은 리다이렉트의 HTTP 메시지의 대화가 증가하므로 그만큼 오버헤드(overhead)가 많지만, 장점도 있다.
- DNS 서버를 세밀하게 설정하는 방법은 클라이언트측의 DNS 서버와 캐시 서버의 거리를 계산하므로 정밀도가 떨어질 수 있지만, 리다이렉트는 클라이언트가 보내는 HTTP 메시지의 송신처 IP 주소를 바탕으로 거리를 판단하므로 정밀도가 높다.
- 경로 정보를 사용하지 않고 다른 정보에서 거리를 계산하여 정밀도를 더욱 높일 수 있다.
- 리다이렉트용 서버가 클라이언트에 반송하는 것은 Location 헤더를 포함한 HTTP 메시지 뿐만이 아니다.
- 패킷의 왕복 시간을 통해 캐시 서버까지의 거리를 계산하여 최적의 캐시 서버에 액세스하도록 스크립트 프로그램을 내장한 페이지를 반송하는 방법도 있다.
- 이 페이지에는 몇 개의 캐시 서버에 시험적으로 패킷을 보내고 왕복 시간을 계측한 후 가장 왕복 시간이 짧은 캐시 서버에 요청 메시지를 다시 보낸다는 내용의 요청 프로그램을 내장해 둔다.
- 이렇게 해서 클라이언트 스스로 최적의 캐시 서버를 판단하고, 여기에 액세스할 수 있다.

5. 콘텐츠 배포 서비스

캐시 내용의 갱신 방법

- 캐시 서버의 효율을 좌우하는 요소 중 캐시의 내용을 갱신하는 방법이 있다.
- 원래 캐시의 개념에는 한 번 액세스한 데이터를 저장해 두고, 이것을 두 번째 이후의 액세스 동작에 이용하여 액세스 동작의 효율을 높이는데 있었지만 이 방법은 최초의 접속 동작에 도움이 되지 않는다.
- 또한 두 번째 이후의 접속에서도 원래 데이터를 가진 웹 서버에 갱신된 내용의 유무를 확인한다는 동작이 있기 때문에 이것이 혼잡하게 뒤얽히면 응답 시간이 악화된다.
- 이 점을 개선하려면 웹 서버에서 원래 데이터를 갱신할 경우 이것을 즉시 캐시 서버에 반영해야 한다.
- 그러면 캐시의 데이터는 항상 최신의 상태를 유지할 수 있으므로 원래 데이터의 갱신을 확인할 필요가 없게 되고, 최초의 접속 동작에도 캐시의 데이터를 이용할 수 있다.
- 콘텐츠 배포 서비스에 이용하는 캐시 서버에는 이러한 대책이 내장되어 있다.

5. 콘텐츠 배포 서비스

캐시 내용의 갱신 방법

- 웹 페이지는 사전에 준비해 두는 정적인 것이 아니라 요청 메시지를 받았을 때 CGI 애플리케이션 등에서 동적으로 페이지를 만드는 경우도 있다.
- 따라서 이러한 것은 캐시 서버에 데이터를 저장해 두면 안된다.
- 이 경우 페이지 전체를 캐시에 저장하지 않고 애플리케이션에서 만든 부분, 즉 매번 페이지의 내용이 달라지는 부분과 달라지지 않는 부분을 구분하고 변하지 않는 부분만 캐시에 저장해야 한다.
- 방화벽이나 프록시 서버, 캐시 서버 등 웹 서버의 바로 앞에는 다양한 서버가 있는데, 요청 메시지는 최종적으로 이곳을 통과하여 웹 서버에 도착한다.
- 웹 서버는 요청 메시지를 받고 요구 내용을 조사하여 요구에 따라 응답 메시지를 만들어 반송한다.

Thank You