컴퓨터 네트워크 – 11 주차

Jong-Kyou Kim, PhD

2015-05-12

Session

- 사용자를 식별하는 일종의 Cookie

```
GET /spec.html HTTP/1.1
Host: www.example.org
Cookie: JSESSIONID=at0nMMEhWhsRkxE5F5XpTF
Accept: */*
```

Session 을 이용한 작업

- /login (Set-Cookie JSESSIONID=at0nMMEhWhsRkxE5F5XpTF)
- /account (Cookie: JSESSIONID=at0nMMEhWhsRkxE5F5XpTF)
- /transfer (Cookie: JSESSIONID=at0nMMEhWhsRkxE5F5XpTF)
- /submit (Cookie: JSESSIONID=at0nMMEhWhsRkxE5F5XpTF)
- /account (Cookie: JSESSIONID=at0nMMEhWhsRkxE5F5XpTF)
- /logout (Set-Cookie JSESSIONID=invalid)

Cookie 를 이용한 세션관리의 한계

- JSESSIONID 값을 다른 사람이 알고 있다면?

  $\longrightarrow$ Session hijacking

- JSESSIONID 를 다른 사람이 추정할 수 있다면?

  $\longrightarrow$ Session hijacking

- submit 이 두번 호출된다면?

  $\longrightarrow$ Application programmer 의 역할

트랜잭션 id를 발급해서 트랜잭션 id가 같은 게 들어오면 버리는 전략을 쓴다!

웹 생태계

- HTTP 에 기반한 서비스의 확산 $\longrightarrow$ 상용화 $\longrightarrow$ 기술의 (단순한 적용에서오는) 한계를 넘기위한 노력 $\longrightarrow$ 다양한 기반 서비스의 출현
- 독자적으로 구축하는 것이 사실상 불가능한 다양한 기반 시설의 활용 지식 $\longrightarrow$ 더 낳은 서비스의 설계/운용
- Caching $\longrightarrow$ Content delivery

전 세계의 컨텐츠를 전세계에 동일한 속도로 전달해주는 서비스

HTTP Caching

- 웹브라우저 – 네트웍을 가능한 효율적으로 활용하려고 한다 저장해두는 곳은 하드디스크
  - 일단 받아둔 데이터 $\longrightarrow$ 다시 활용한다
  - If-Modified-Since 는 날자만 확인. 충분한가?
- 언제 갱신해야 하나? $\longrightarrow$ Expires 헤더

HTTP Caching

```
HTTP/1.1 200 OK
Date: Fri, 30 Oct 1998 13:19:41 GMT
Server: Apache/1.3.3 (Unix)
Cache-Control: max-age=3600, must-revalidate
Expires: Fri, 30 Oct 1998 14:19:41 GMT
Last-Modified: Mon, 29 Jun 1998 02:28:12 GMT
ETag: "3e86-410-3596fbbc"
Content-Length: 1040
Content-Type: text/html
```

이전에 가져갔을 때 이 코드를 받았다~
혹시 이 코드가 변했니?
이 값이 일치하면, 캐시 그대로 쓸게!
캐시가 만료되어도 쓴다.

- 쿠키를 끄는 사람들한테도 쿠키 대신 쓸 수 있도록…
쿠키 대신에 unique하게 쓸 수 있다!

Jong-Kyou Kim, PhD

Web Proxy Cache

네트워크 프로그램에서 성능을 올리는 가장 좋은 방법은
네트워크를 쓰지 않도록 하는 방법이다!
로컬을 이용하게 하는 방법!!
네트워크를 굳이 써야한다면 한번에 가능한 많이 보내는 게 좋다.
(한번 보낼때마다 3-way handshaking해야 하잖아... - 이 때마다 latency!)
커넥션을 한번열고 닫지 않기~
굉장히 많은 데이터를 보낸다고 한다면, 슬로우 스타트를 고려하면 여러 커넥션을 유지!

- 사용자별로 설정
- 모든 Request 는 proxy 로 전달
  - proxy 에 cache 된 내용 $\longrightarrow$ 서버에 접근하지 않고 전달
  - 서버에서 가져온 내용 $\longrightarrow$ proxy 에 저장

지구 반대편에서 가져오지 않고 내 바로 옆에서 가져오는 거지~
한명이 지구반대편에서 가져오면, 다른 애들은 개꺼에서 가져오는 거지!!

네트워크에서 중요한 상황은 에러가 날때다!
서버는 살아있는데(네트워크 프로토콜은 연결) 어플리케이션 연결이 안돼!
ping을 보내면 알 수 있겠지.
이런 경우 어떻게 해야할까?

모니터링 프로그램
웹어플리케이션이 잘 살아있나 계속 체크하다
죽었으면 다시 띄워~
로컬에서 네트워크로 신호를 보내봐
웹어플리케이션이 살아있다면 무조건 도착해야지.
=> 127.0.0.1(loop back)
- 데이터링크와 물리 레이어까지는 가지 않고
패킷이 도착한다!

Web Proxy Cache

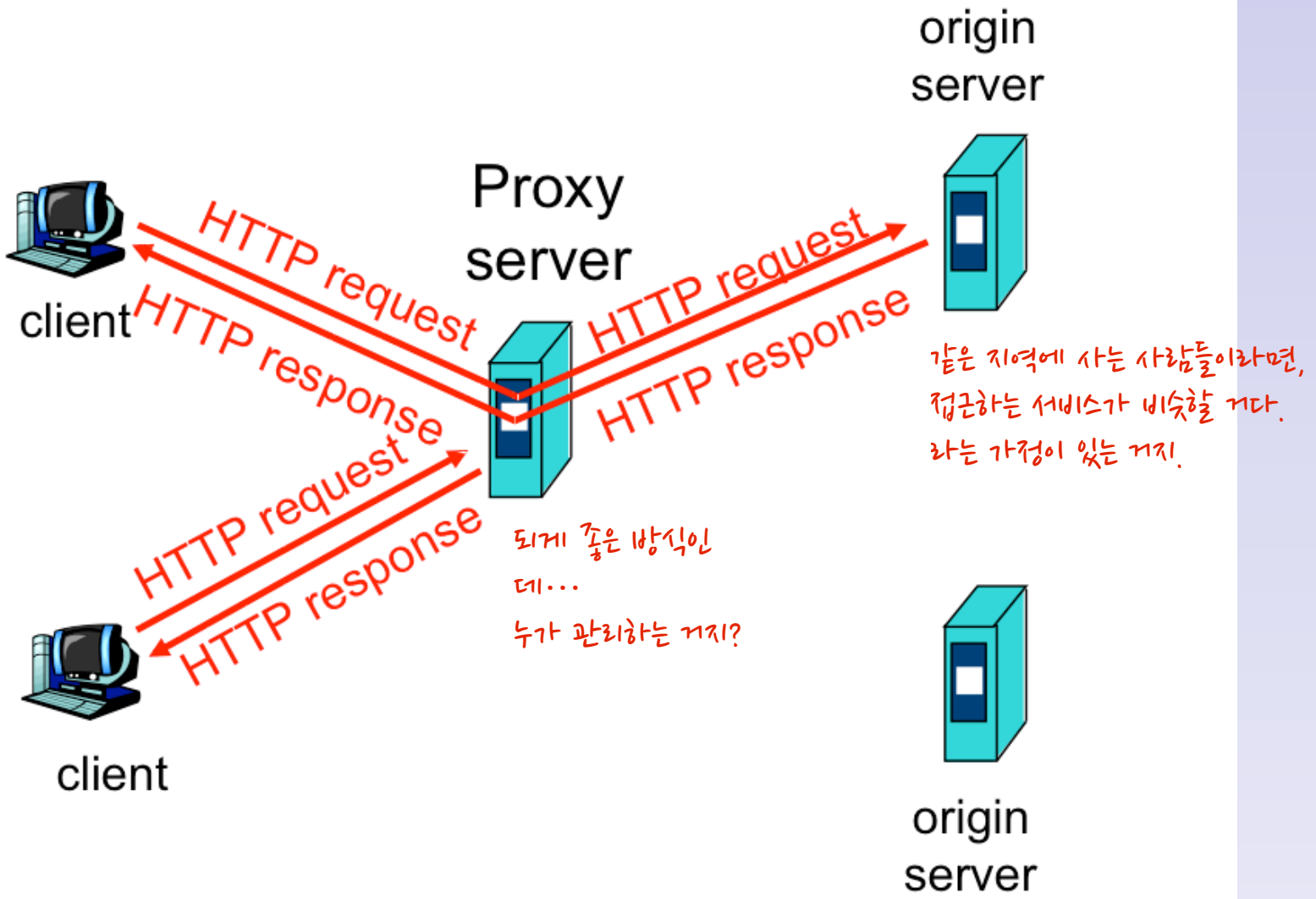

그림: Web proxy

Jong-Kyou Kim, PhD

웹트래픽의 분석

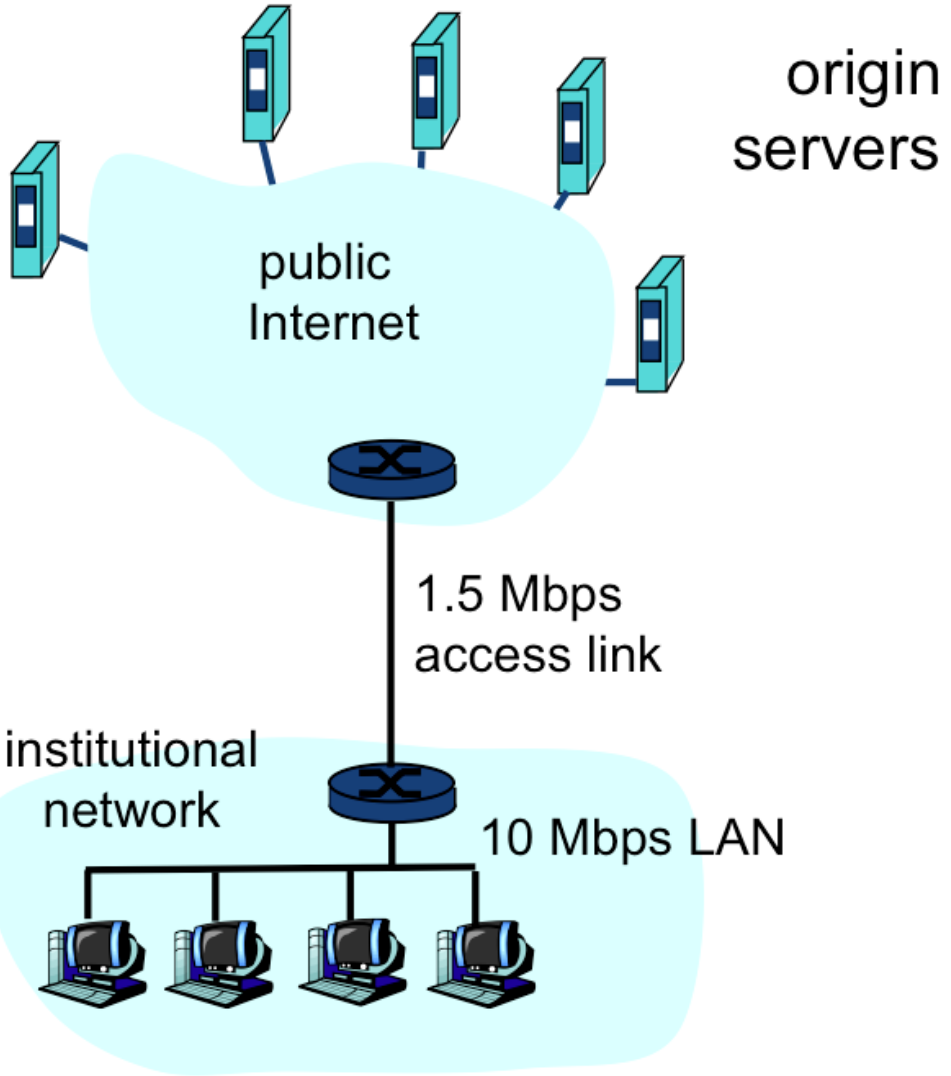

그림: Web traffic

Jong-Kyou Kim, PhD

웹트래픽의 분석

- 가정
  - 인터넷 회선용량: 1.5Mbps
  - 평균 파일크기 = 100,000 bits (10 KB) 보통 웹페이지들이 이정도 크기다!
  - 초당 약 15 회씩 request
  - 인터넷을 통과하는데 걸리는 지연 2 초
- 분석
  - 내부망 활용률 = 15%
  - 외부망 활용률 = 100% congestion
  - 전체 지연시간 = 인터넷지연 + LAN = 약 2 초 + 몇 분

congetsion 이 생기면...

회선 증설 효과

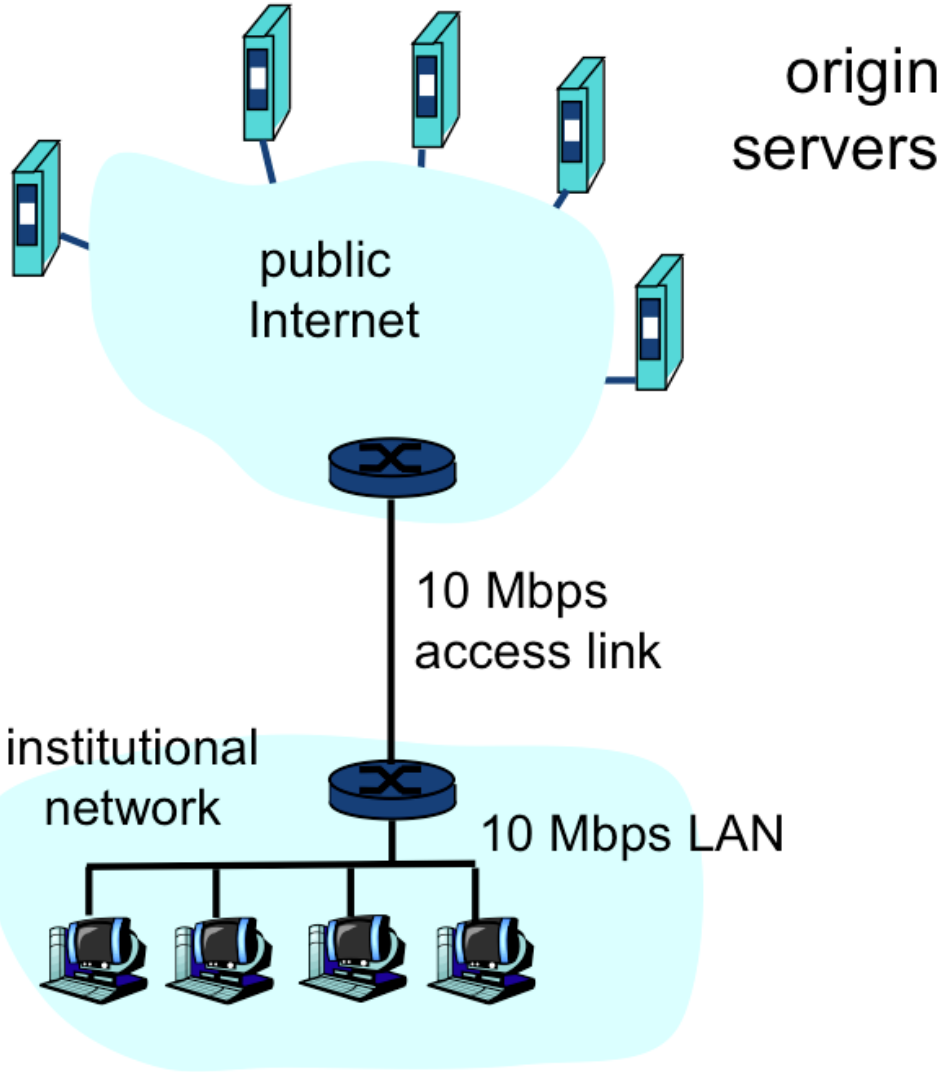

그림: 회선증설

Jong-Kyou Kim, PhD

회선 증설 효과

- 가정
  - 인터넷 회선용량: 10Mbps
  - 평균 파일크기 = 100,000 bits (10 KB)
  - 초당 약 15 회씩 request
  - 인터넷을 통과하는데 걸리는 지연 2 초
- 분석
  - 내부망 활용률 = 15%
  - 외부망 활용률 = 15% $\longrightarrow$ no congestion
  - 전체 지연시간 = 인터넷지연 + LAN = 약 2 초

캐쉬 설치효과

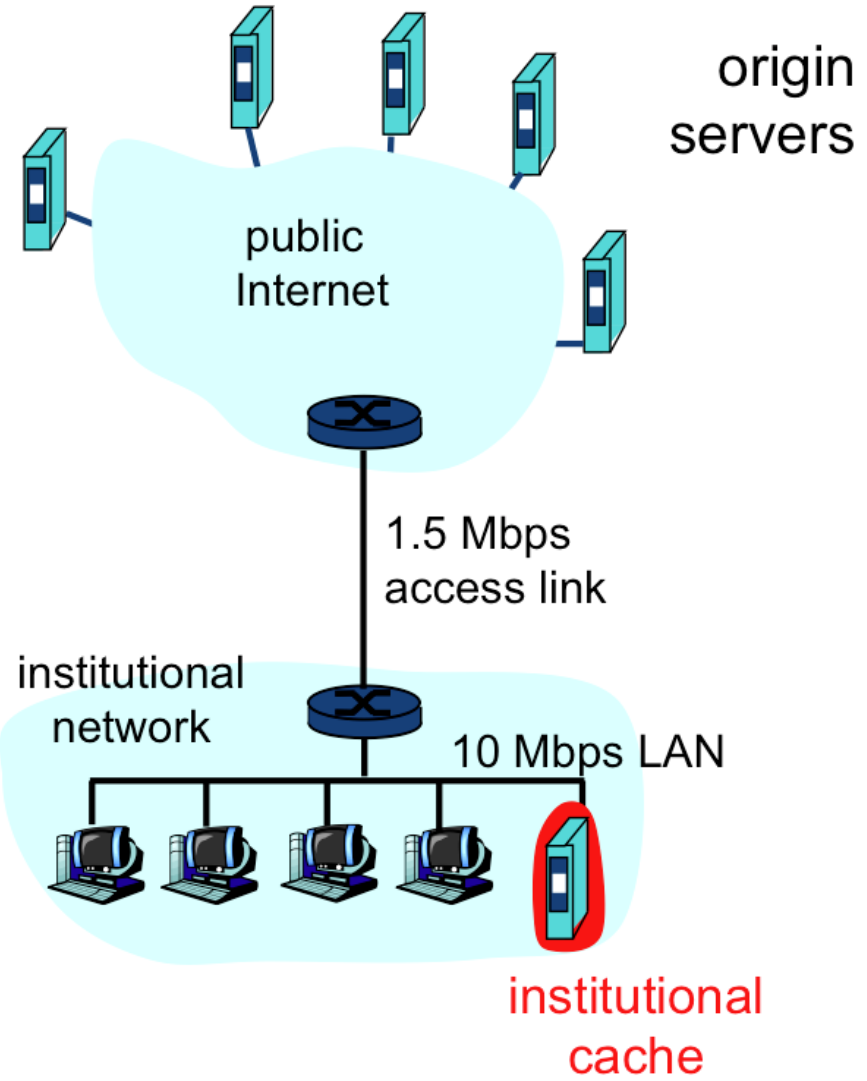

그림: 캐쉬설치

캐쉬 설치효과

- 가정
  - 인터넷 회선용량: 1.5Mbps
  - Hit ratio: 0.4 40% 한번 가져온 파일을 누군가 다시 볼 확률
  - 평균 파일크기 = 100,000 bits (10 KB)
  - 초당 약 15 회씩 request
  - 인터넷을 통과하는데 걸리는 지연 2 초
- 분석
  - 내부망 활용률 = 15%
  - 외부망 활용률 = 60% (40% 는 내부에서 처리)
  - 전체 지연시간 = (인터넷지연*0.6) + LAN = 약 1.2 초

인터넷 회선을 바꾸는 건
비용이 거의 10배를 더 지불해야 한다.
서버하나 설치하는 것이 더 효율적이네!

캐쉬의 한계

- 50% 이상의 HTTP 개체는 cache 가 불가능하다
  - 동적으로 변하는 데이터: 주식가격, 운동경기 결과, 웹캠
  - 암호화된 데이터 (HTTPS)
  - Cookie $\longrightarrow$ 남의 세션에 들어가게 됨!
  - Hit metering $\longrightarrow$ 광고비 산정의 기준

하지만 그냥 버리기엔 너무 아깝잖니~

Jong-Kyou Kim, PhD

CDN and server selection

글로벌하게 프록시 서버의 인프라를 구축해서 돈을 받는 거지~~!

Jong-Kyou Kim, PhD

- 많은 서버에 내용을 복사해 두는 것
- 일종의 캐쉬 $\longrightarrow$ Latency 감소
  - 최신의 내용
  - 최적의 장소

  $\longrightarrow$ 어떻게? (HTTP 기반 + DNS 기반)
- 클라이언트가 요청한 자료를 다른 서버에서 찾아가도록 구현

  $\longrightarrow$ redirection

CDN and server selection

- 서비스의 품질을 높이는데 중요한 역할을 함
  - 광대역 통신망의 보급 $\longrightarrow$ Latency 가 품질

$\longrightarrow$ 빛의 속도에 제한 $\longrightarrow$ 캐쉬가 유일한 해결책

가까운 지역에 갖다놓는 게 유일한 해결책

최선의 서버 선정

- 지역적으로 가까운 서버 $\longrightarrow$ 전화국 옆에 사는 LGT 가입자는?
- RTT 가 가장 작은 서버 $\longrightarrow$ Packet loss 가 많이 발생한다면?
- Packet loss 가 작은 서버 $\longrightarrow$ Congestion 이 발생했다면?

$\longrightarrow$ 어려운 문제

CDN 구현 방식 (HTTP 방식)

- Redirection 이용 (30x response)
- 작동방식
  - 서버가 get request 를 받는다 본점에서 손님을 받는다. 본점에서 자기가 처리할지 분점으로 보낼지 결정
  - 해당 요청을 처리할 적절한 서버를 찾는다 load balancing
  - Redirect response 를 보낸다

CDN 구현 방식 (HTTP 방식)

- 장점
  - 구현이 간단하다
  - 서비스 상황을 가장 잘 알고 판단을 내릴 수 있다
- 단점
  - 일단 확보한 연결을 계속 사용할 수 없다

$\longrightarrow$ latency 를 증가시킬 수 있다

CDN 구현방식 (DNS 방식)

- DNS 에서 반환하는 IP 로 조절 그래서 요즘엔 DNS를 이용해서 load balancing을 한다!
- 작동방식
  - 클라이언트가 IP 를 요청한다
  - 가장 빨리 응답할 수 있는 서버의 IP 주소를 반환한다

CDN 구현방식 (DNS 방식)

- 장점
  - 처음부터 서비스할 서버로 연결해 준다 커넥션이 유지됨!
  - 서비스 개발과 분리하여 운영할 수 있다
- 단점
  - 네트웍에 대한 이해가 필요하다 $\longrightarrow$ (설정이 어렵다)
  - 서버의 정보를 DNS 와 공유해야 한다 어느 서버가 붐비고, 어느 서버가 널럴한지 정보를 공유해야함

Akamai

Jong-Kyou Kim, PhD

- 세계에서 가장 큰 CDN
- 최초로 CDN 의 개념을 소개함 $\longrightarrow$ CDN 의 원형
- 작동 방식
  - 이미지 등 정적인 자료만 서비스를 제공함
  - html 내의 link $\longrightarrow$ akamai link 로 변환됨 모든 데이터를 akamai에서 가져오는 것처럼
  - 캐쉬된 자료 $\longrightarrow$ 바로 서비스함
  - 캐쉬되지 않은 자료 $\longrightarrow$ 적절한 서버를 찾아 처리
- 회사로서의 Akamai (2011)
  - 매출 : $1.16B
  - 수익 : $290.65M
  - 종업원: 3000 명

Akamai 의 동작방식

- 전 세계에 서버를 설치 학교에 놓는 컴퓨터가 매우 많았다! 해당 지역에 있는 대학교를 선택해서 지원받음!
- 고객이 요청한 자료를 복사
- 고객 서버에 자료 요청
  - 예: cnn.com/index.html
- 복사된 자료 부분은 자동으로 변경
  - 예: `<img src="http://cnn.com/af/x.gif">`

$\longrightarrow$

```
<img src="http://a73.g.akamaitech.net/7/23/cnn.com/
```

- Akamai DNS 가 적절한 서버로 요청을 전달

Akamai 의 동작방식

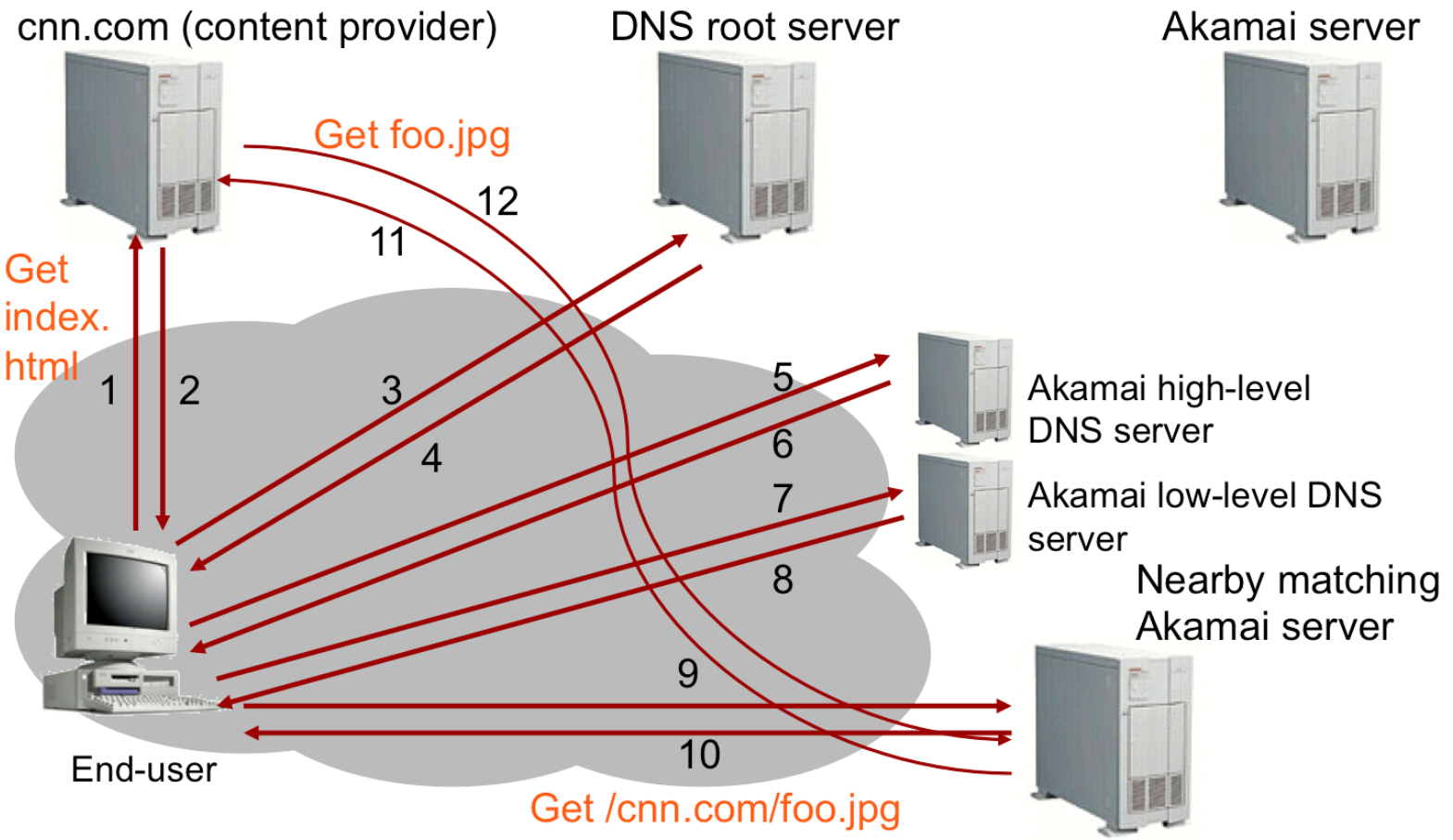

그림: Akamai 동작방식

Akamai 의 동작방식

- DNS 제어
  - Query $\longrightarrow$ akamai.net
  - akamai.net $\longrightarrow$ g.akamaitech.net
    - TTL 은 큰 값으로 설정됨 $\longrightarrow$ 왜? 거의 변할 일이 없으니까
  - g.akamaitech.net $\longrightarrow$ 지역서버 선정
    - TTL 은 작은 값으로 설정됨 $\longrightarrow$ 왜? 상황에 따라 민첩하게 대응해야 하니까

TTL 이 뭐였지?
time to live
DNS에서 한시간동안은 그대로 써~

HTTP 요약 – 1

Jong-Kyou Kim, PhD

- 간단한 텍스트 기반의 프로토콜
  - 응답시 status/error 뿐 아니라 authentication, client 측면의 state 관리, cache 관리 등을 포함함
- HTML 지원을 위해 고안됨
  - 오늘날 가장 널리 사용되는 인터넷 자료 형식
- 서버쪽에 부담을 많이 줄 수있는 구조라는 한계

http가!

1.1부터는 많이 개선이 되었다~
2.0은 더 많이 개선되었는데
데이터링크의 알고리즘이 많이 들어가있다.

HTTP 요약 – 2

- TCP 와의 관계
  - TCP 의 stream 기능에 의존
  - 서버의 부담은 persistent connection 으로 대응
- 성능향상
  - persistent connection
  - pipeline
  - caching
  - replication CDN 정적인 자료들을 복사해서~