Wireshark를 이용한 Application Layer Protocol 트래픽 분석

Domain Name Service

제출자
201323148
소프트웨어학과
이제호

목차

1. 관찰 목표 및 예상 결과

1-(1) 관찰 목표

DNS는 사용자가 HTTP, FTP, SMTP 등의 애플리케이션 서비스를 이용할 때 사용자가 제공한 도메인 네임을 IP주소로 변환하는 서비스를 말한다. 사용자의 눈에 드러나게 직접 상호작용하는 애플리케이션이 아니라 중요성을 실감하기 어렵지만, DNS가 없다면 핵심적인 인터넷 기능을 서비스하는 시스템이 무너지므로 인터넷 상의 모든 애플리케이션 서비스를 이용하지 못한다. 비교하자면 우편 시스템에서 우체국이 없어지는 것이다. 전 세계의 모든 우편이 하나의 우체국을 거치지 않듯이, 모든 DNS message가 하나의 DNS server를 거치지 않는다. DNS는 분산된 서버로 구현되어 각 호스트가 분산된 서버들 중 하나로 query를 보내도록 하는 애플리케이션 계층 프로토콜이고, 주소 매핑 만을 목적으로 하는 거대한 시스템이다. 단순한 작업을 위해 세계적인 분산DB가 구축되어 있고 그 속에 정책적, 사업적인 요소가 복잡하게 얽혀있다는 점은 DNS가 어떻게 작동하는지 관찰할 동기부여가 됐다.

DNS는 사용자가 직접 상호작용하는 애플리케이션이 아니고 보통 1개의 query 메시지와 1개의 response 메시지만 발생하기 때문에 traffic의 흐름을 분석하기엔 적합하지 않다.

따라서 본 보고서에서는 DNS 서비스를 이용할 때 어떤 protocol 메시지를 송·수신하는지 알아보기 위해, wireshark로 protocol 메시지를 분석하여 DNS 서비스의 기본적인 기능을 살펴보고 콘솔 명령 프로그램을 이용해 부가적으로 내용을 확인하여 서술하는 것을 목표로 정했다.

1-(2) 관찰 개요

호스트 관점에서 DNS 서비스를 볼 때, DNS 서비스 이용 과정을 두 가지 Process로 나눠봤다. 첫 번째는 mapping process이고 두 번째는 caching process이다. Mapping process는 사용자에 의해 브라우저에 입력된 특정 domain name에 해당하는 ip주소를 얻는 과정이고, caching process는 mapping이 끝난 domain name에 대하여 발생하는 같은 query를 빠르게 처리하기 위해 cache memory에 mapping 정보를 저장함으로써 네트워크의 DNS 메시지 수를 줄이는 과정이다. 첫 줄에서 언급했듯이, '호스트 관점'에서의 DNS caching은 해당 사용자 호스트 Operating System의 메모리에 mapping 정보를 저장하는 것을 의미한다. DNS server 관점의 DNS caching은 호스트가 local DNS server에게 요청 메시지를 보내고 해당 서버가 mapping 정보를 얻게 되면 server의 cache memory에 일정 시간 저장해두고 같은 요청에 대해 빠르게 응답하는 과정을 말한다.

1-(3) 예상 결과

위에서 언급한 DNS를 사용하는 두 개의 process를 간단한 단계의 웹 브라우징을 통해 관찰하도록 할 것이다. 우선 mapping process 관찰을 위해 브라우저 주소 창에 특정 도메인 네임을 입력하고 해당 페이지로 이동한 뒤 메인 페이지의 콘텐츠를 클릭하여 로딩하는 과정을 거칠 것이다. 그 결과 두 번의 페이지 이동에서 2개의 query 메시지와 2개의 reponse 메시지가 나타날 것으로 예상된다. query 메시지에는 도메인 네임과 query 정보가 있을 것 이고 response 메시지에는 매핑한 ip 주소가 있을 것 이다. 이 과정에서 발생하는 대부분의 application layer packet traffic은 해당 콘텐츠를 로딩하는 HTTP 메시지일 것이다. 콘텐츠 페이지 로딩이 완료되면 DNS caching process를 관찰하기 위해 새로고침(같은 페이지를 요청)을 해서 어떤 메시지가 발생하는지 볼 것이다. 그 결과로 웹 브라우저는 OS의 cache에서 바로 ip주소를 얻어올 것이고, DNS client는 어떤 메시지도 송·수신하지 않을 것으로 예상된다.

이렇게 웹 페이지 이동, 로딩, 갱신하는 사용자의 세가지 행위(주소 창 입력, 링크 버튼 클릭, 새로고침)를 통해 발생하는 DNS 메시지들을 살펴볼 것이다.

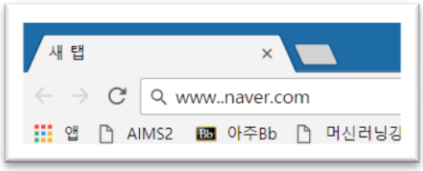
2. 관찰 방법

DNS가 제공하는 기본적인 두 개의 service process를 간단한 단계의 웹 브라우징을 통해 관찰하도록 할 것이다.

본인의 노트북으로 아주대학교 중앙도서관 열람실에서 관찰할 것이고 Wireshark 프로그램을 이용하여 packet traffic을 살펴보고 각각의 DNS 메시지들을 자세히 살펴볼 것이다. Traffic의 흐름을 위주로 관찰하는 대신, 각 수행 단계에서 볼 수 있는 DNS 메시지의 발생 유무와 각 메시지의 자세한 내용을 살펴봄으로써 분석을 할 것이다. 각 수행 과정이 모두 끝나면 캡처를 중지하고 filter 값을 "dns or http"로 입력하고 DNS, HTTP 메시지를 관찰한다. 특히 DNS 메시지의 구조와 각각의 내용에 대해 살펴보고, 모든 DNS, HTTP 메시지를 I/O Graph 기능을 이용해 도표화시킨다. HTTP 메시지를 송·수신하기 전에 연결을 설정하는 TCP 메시지는 transport layer와 관련된 메시지이기 때문에 필터 값으로 추가하지 않았다.

구체적인 관찰 과정은 다음과 같다.

(1) Naver 홈페이지 이동

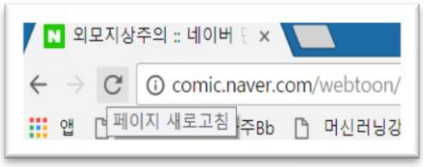


우선 Wireshark 프로그램의 start capturing packets 버튼을 클릭하여 캡처를 시작한다. 그 다음 크롬 브라우저의 주소창에 네이버의 도메인네임인 www.naver.com을 입력하고 엔터를 클릭하여 네이버 메인 페이지로 이동한다.

(2) 웹툰 페이지 이동

네이버 메인 페이지에 보이는 '외모지상주의' 웹툰 바로가기를 클릭한다. (당시 웹툰 메인 이미지에 박태준 작가의 '외모지상주의' 178화로 이동하는 링크가 업데이트 되어 있었다)

(3) 웹툰 페이지 재방문(새로고침)



웹툰 페이지 로딩이 끝난 후, 같은 페이지를 다시 요청하기 위해 새로고침 버튼을 클릭한다. 이 과정은 OS DNS caching이 발생했는지 확인하기 위한 과정이다.

(4) DNS cache data 삭제 후 웹툰 페이지 재방문(새로고침)

OS의 DNS cache에 있는 정보를 모두 삭제하기 위해 윈도우 콘솔창에 명령어 `ipconfig/flushdns` 를 입력한다. 또한 크롬 브라우저에 저장된 DNS cache data를 삭제하기 위해 크롬 설정에서 인터넷 사용기록을 모두 삭제한다. 모든 cache data 삭제 과정이 끝나면 (3)번 과정을 다시 수행(새로고침)하여 같은 페이지를 한번 더 요청한 후에 패킷 캡처를 중지한다.

3. Traffic 분석

관찰 장소 : 아주대학교 중앙도서관 2층 커뮤니티 라운지

관찰 기간 : 약 40초

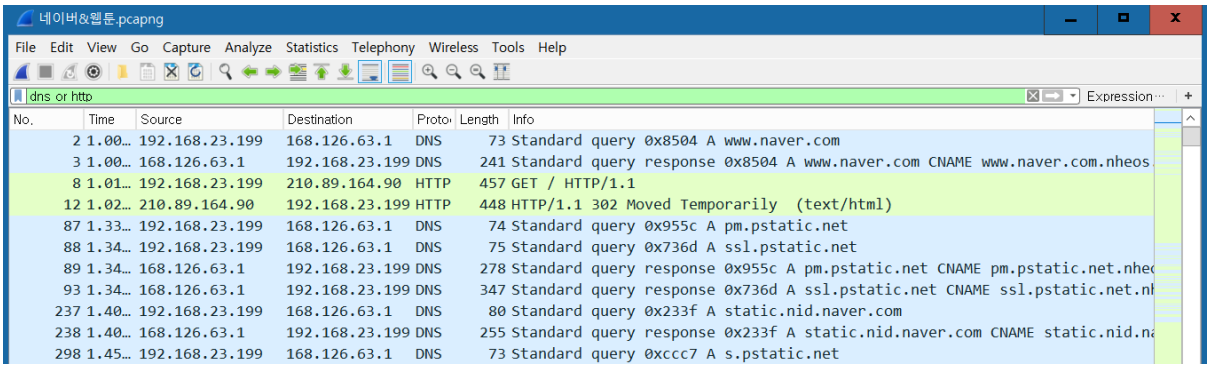
Application Layer : **DNS**, HTTP

<Protocol Hierarchy>

Frame	100.0	914	100.0	472212	103 k	0
Ethernet	100.0	914	2.7	12796	2795	0
Internet Protocol Version 4	100.0	914	3.9	18280	3993	0
User Datagram Protocol	22.3	204	0.3	1632	356	0
Domain Name System	22.3	204	5.5	25768	5629	204
Transmission Control Protocol	77.7	710	87.6	413722	90 k	0
Hypertext Transfer Protocol	77.7	710	3743.9	17678972	3862 k	355

Statistics 탭의 Protocol Hierarchy 기능으로 계층적 구조를 살펴보면 위 사진과 같다.
IPv4의 상위 프로토콜인 UDP와 TCP가 존재하였고 본 관찰에서 발생한 모든 UDP 세그먼트는 DNS 서비스로부터 발생하였으며 모든 TCP 세그먼트는 HTTP 서비스로부터 발생하였다. 위 구조를 통해 DNS 서비스는 UDP/IP 기반, HTTP 서비스는 TCP/IP 기반의 Application Layer Service임을 알 수 있다.
아래 (1)번의 사진에서 보이듯이 캡처를 시작하고www.naver.com을 입력하자 마자 두 개의 DNS 패킷(No.2, No.3)이 발생하는 것을 확인할 수 있는데, 이는 웹툰 페이지를 요청할 때 보이는 HTTP 패킷 발생 과정과는 다른 것을 알 수 있다. 예를 들어 웹툰 페이지를 요청하고 첫 HTTP 패킷이 발생하기 전, TCP handshaking을 위한 메시지가 TCP 연결을 설정하고 나서야 첫 HTTP 패킷이 발생하는 것을 확인하였다. 반면에 UDP는 연결 설정 과정이 필요 없기 때문에, TCP 보다 빠른 전송이 가능하다. 따라서 적은 수의 메시지만 필요하고 빠른 메시지 송·수신이 필요한 DNS 서비스는 UDP에서 구현하는 것이 적합하다. DNS 서비스는 UDP 포트번호 53을 이용한다.

(1) Naver 홈페이지 이동 – packet No. 2 ~ No. 12



처음으로 NAVER의 메인 홈페이지를 이동했을 때 첫 2개의 패킷은 DNS 메시지로, 호스트 Client로부터 168.126.63.1로 query 메시지를 보냈고, 그 반대방향으로 response 메시지를 받았다. 네트워크 속성을 보면 168.126.63.1은 IPv4 DNS 서버임을 알 수 있는데, nslookup을 실행시키면

옆의 사진과 같다. 즉, 내 호스트 DNS Client는 항상 kns.kornet.net의 이름을 가진 DNS 네임서버(KT 소유)로 DNS query 메시지를 보내고, kns.kornet.net은 local DNS server이다.
이제 DNS 메시지를 자세히 살펴보기 위해 각 메시지를 열어보면 다음 사진과 같다.

```
> User Datagram Protocol, Src Port: 63871, Dst Port: 53
v Domain Name System (query)
    [Response In: 3]
    Transaction ID: 0x8504
  v Flags: 0x0100 Standard query
      0... .... .... .... = Response: Message is a query
      .000 0... .... .... = Opcode: Standard query (0)
      .... ..0. .... .... = Truncated: Message is not truncated
      .... ...1 .... .... = Recursion desired: Do query recursively
      .... .... .0.. .... = Z: reserved (0)
      .... .... ...0 .... = Non-authenticated data: Unacceptable
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  v Queries
    > www.naver.com: type A, class IN
```

DNS query 메시지

위 사진은 DNS query 메시지로 message format을 살펴보자.

Transaction ID	처음 필드인 Transaction ID 필드는 response 메시지의 Transaction ID 값과 같고, 식별하는 번호이다.
Flags	Flags는 control 필드로 위 메시지는 0x0100 플래그 값을 가진 standard query이다. 플래그는 기본적으로 질의 또는 응답 메시지인지 결정해주고 name resolution의 방식으로 recursive query를 하도록 요청하는 등의 16비트로 구성되어 있다.
Questions ~AdditionalRRs	메시지에 포함된 각 항목의 수를 나타낸다.
Queries	현재 query에 대한 정보를 포함하는 필드로 domain name field와 type field가 있다. Type은 A(ipv4 주소 매핑), NS(authoritative 서버 정보 획득), CNAME(canonical name 획득), MX(메일 서버 정보 획득) 가 있고, 각 타입 레코드가 queries 필드에 포함되어 전달되면 서버 측은 어떤 응답을 해줄지 구분한다.

```
v Domain Name System (response)
    [Request In: 2]
    [Time: 0.006585000 seconds]
    Transaction ID: 0x8504
  v Flags: 0x8180 Standard query response, No error
      1... .... .... .... = Response: Message is a response
      .000 0... .... .... = Opcode: Standard query (0)
      .... .0.. .... .... = Authoritative: Server is not an authority for domain
      .... ..0. .... .... = Truncated: Message is not truncated
      .... ...1 .... .... = Recursion desired: Do query recursively
      .... .... 1... .... = Recursion available: Server can do recursive queries
      .... .... .0.. .... = Z: reserved (0)
      .... .... ..0. .... = Answer authenticated: Answer/authority portion was not authenticated by the server
      .... .... ...0 .... = Non-authenticated data: Unacceptable
      .... .... .... 0000 = Reply code: No error (0)
    Questions: 1
    Answer RRs: 3
    Authority RRs: 3
    Additional RRs: 3
  v Queries
    > www.naver.com: type A, class IN
  v Answers
    > www.naver.com: type CNAME, class IN, cname www.naver.com.nheos.com
    > www.naver.com.nheos.com: type A, class IN, addr 210.89.164.90
    > www.naver.com.nheos.com: type A, class IN, addr 125.209.222.142
  v Authoritative nameservers
    > nheos.com: type NS, class IN, ns ns1.nheos.com
    > nheos.com: type NS, class IN, ns ns2.nheos.com
    > nheos.com: type NS, class IN, ns ns3.nheos.com
  v Additional records
    > ns1.nheos.com: type A, class IN, addr 125.209.216.100
    > ns2.nheos.com: type A, class IN, addr 61.247.202.53
    > ns3.nheos.com: type A, class IN, addr 175.158.30.70
```

DNS response 메시지

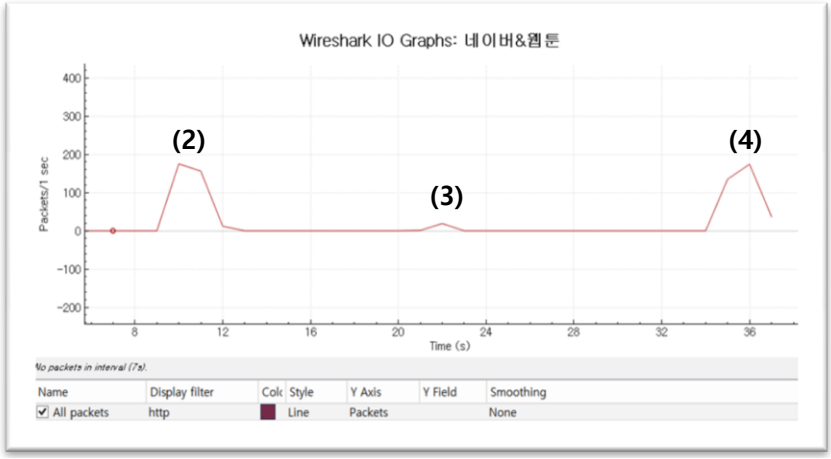
위 사진은 DNS response 메시지로 message format은 query 메시지와 같다.

Flags	0x8180은 에러없는 standard query에 대한 응답을 나타낸다. Query 메시지에서는 사용하지 않는 flag의 마지막 4비트는 reply code로 현재 0000인 no error를 나타낸다. 그리고 Answer authenticated에 해당하는 1개짜리 비트는 응답해주는 ip주소 매핑 정보를 해당 서버가 책임지고 저장하고 있는지에 대한 정보인데, 현재 0이므로 최종 응답해주는 서버는 네이버에 대한 ip주소 정보를 저장하고 있는 서버가 아니다.
Answers	Answers 필드는 질의에 대한 최종 응답 정보들을 포함한다.
Authoritative nameservers	www.naver.com 에 대한 ip주소를 저장하고 있는 Authoritative nameserver의 이름을 포함한다.
Additional records	Additional records 필드는 위 Authoritative 서버들의 ip주소를 포함한다.

Response 메시지에서 Answers 필드를 자세히 볼 필요가 있는데, 총 3개의 답변이 온 것을 확인할 수 있다. 우선 DNS 서버는 www.naver.com이 사실은 네이버의 실제 domain name이 아니고, www.naver.com.nheos.com이 실제 domain name임을 알려주는 응답을 했다(type CNAME). 또한 네이버의 실제 domain name에 대한 2개의 ip 매핑 주소를 넘겨주었는데, 2개 모두 www.naver.com.nheos.com에 대한 ip 주소이다(type A).

DNS Client로부터 www.naver.com.nheos.com의 ip주소를 얻은 웹 브라우저는 해당 ip주소의 서버로 HTTP 요청을 할 수 있게 되었다(packet No. 8, No. 12). 8번째 패킷을 보면 '/' URL을 요청하는데 Host: www.naver.com 이므로 www.naver.com/ URL을 요청하는 패킷이다. 다음 패킷은 HTTP 응답으로 302 Moved Temporarily의 상태 코드가 포함되어 보내졌다. 검색해보니 302는 redirected page에 대한 내용으로 정확한 내용은 찾지 못하였지만 정상적으로 네이버 메인 홈페이지를 띄울 수 있었다.

HTTP packet traffic



아래에서 설명할 (2), (3), (4)번 과정에서 발생한 HTTP 패킷 트래픽에 대한 graph이다.
DNS 패킷의 수는 traffic에 영향을 거의 안 주기 때문에 Packet traffic filter 값에 DNS 패킷을 제외하였다.

(2) 웹툰 페이지 이동 – packet No. 3230 ~ No. 11847

```
  3230 10.0… 192.168.23.199   168.126.63.1    DNS      75 Standard query 0x8736 A comic.naver.com
  3231 10.0… 168.126.63.1     192.168.23.199 DNS     272 Standard query response 0x8736 A comic.naver.com CNAME…
  3237 10.0… 192.168.23.199   210.89.168.33  HTTP    649 GET /webtoon/detail.nhn?titleId=641253&no=178 HTTP/1.1
  3262 10.0… 210.89.168.33    192.168.23.199 HTTP    639 HTTP/1.1 200 OK  (text/html)
  3263 10.1… 192.168.23.199   210.89.168.33  HTTP    665 GET /css/comic/common_20180410181119.css HTTP/1.1
  3266 10.1… 192.168.23.199   210.89.168.33  HTTP    665 GET /css/comic/likeit_20180410181119.css HTTP/1.1
  3284 10.1… 210.89.168.33    192.168.23.199 HTTP    226 HTTP/1.1 200 OK  (text/css)
  3289 10.1… 210.89.168.33    192.168.23.199 HTTP    894 HTTP/1.1 200 OK  (text/css)
                                                          ⋮
  3593 10.3… 192.168.23.199   183.111.251.… HTTP    539 GET /webtoon/641253/178/20180322210446_5440bdb495d6830…
  3597 10.3… 192.168.23.199   183.111.251.… HTTP    539 GET /webtoon/641253/178/20180322210446_5440bdb495d6830…
  3599 10.3… 192.168.23.199   183.111.251.… HTTP    539 GET /webtoon/641253/178/20180322210446_5440bdb495d6830…
                                                          ⋮
  4274 10.4… 183.111.251.132  192.168.23.199 HTTP     95 HTTP/1.1 200 OK  (JPEG JFIF image)
  4321 10.4… 183.111.251.132  192.168.23.199 HTTP    514 HTTP/1.1 200 OK  (JPEG JFIF image)
  4360 10.4… 183.111.251.132  192.168.23.199 HTTP   1417 HTTP/1.1 200 OK  (JPEG JFIF image)
                                                          ⋮
 11825 12.3… 192.168.23.199   182.162.92.32  HTTP    485 GET /comic/comment/sp_cbox.png HTTP/1.1
 11847 12.3… 182.162.92.32    192.168.23.199 HTTP    541 HTTP/1.1 200 OK  (PNG)
```

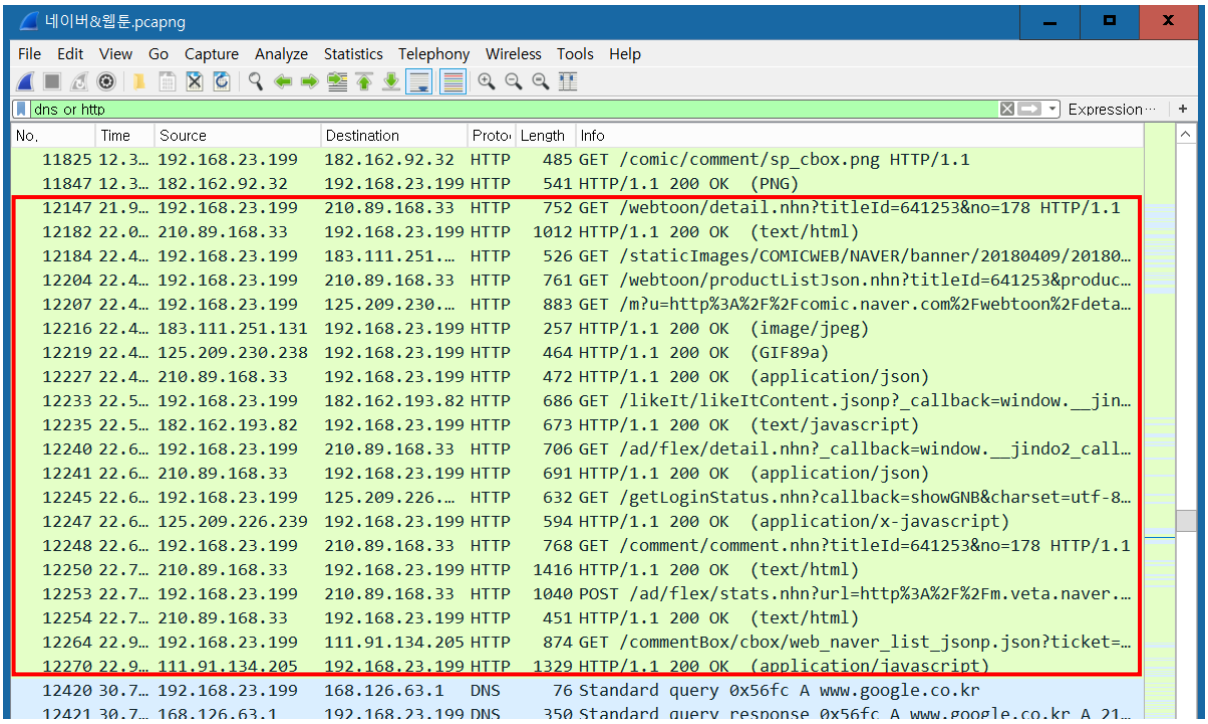
위 사진은 네이버 메인 홈페이지로 이동한 후, 10초 쯤에 웹툰 바로가기를 클릭하고 웹툰 페이지 로딩이 완료될 때까지의 패킷을 나타낸다. 3230번부터 11847번까지의 패킷 중에 DNS 패킷은 단 두 개로 comic.naver.com에 대한 ip주소 질의와 응답 메시지이다. 즉, 모든 웹툰 데이터는 comic.naver.com의 도메인 네임을 가진 서버에 저장되어 있음을 알 수 있다.

```
▾ Domain Name System (response)
    [Request In: 3230]
    [Time: 0.002550000 seconds]
    Transaction ID: 0x8736
  › Flags: 0x8180 Standard query response, No error
    Questions: 1
    Answer RRs: 4
    Authority RRs: 3
    Additional RRs: 3
  › Queries
  ▾ Answers
    › comic.naver.com: type CNAME, class IN, cname comic.naver.com.nheos.com
    › comic.naver.com.nheos.com: type CNAME, class IN, cname comic.nfront.nheos.com
    › comic.nfront.nheos.com: type A, class IN, addr 210.89.168.33
    › comic.nfront.nheos.com: type A, class IN, addr 210.89.168.65
  ▾ Authoritative nameservers
    › nheos.com: type NS, class IN, ns ns3.nheos.com
    › nheos.com: type NS, class IN, ns ns1.nheos.com
    › nheos.com: type NS, class IN, ns ns2.nheos.com
  › Additional records
```

DNS response 메시지를 보면, comic.naver.com의 최종 canonical name은 comic.nfront.nheos.com이고 해당 ip주소를 2개 얻어왔음을 알 수 있다. 추가적으로 www.naver.com.**nheos.com**과 comic.nfront.**nheos.com**의 매핑 정보는 같은 Authoritative nameserver에 저장되어 있는 것을 알 수 있다.

3237번 패킷을 보면 웹 브라우저는 얻어온 ip주소를 목적지로 하여 클릭한 웹툰 페이지의 URL을 요청하고 있다. 3262번 패킷은 200 Ok 상태 코드를 가지고 있고 html파일을 포함한 HTTP 응답 메시지이다. 이후 많은 css, image 파일 등을 요청하며 웹툰 페이지 로딩을 완료한다. 이 과정에서 보이는 거의 모든 패킷은 HTTP 패킷으로 로딩이 완료되고 더 이상 발생하지 않았다(graph 참고).

(3) 웹툰 페이지 재방문(새로고침) – packet No. 12147 ~ No. 12270



(12.3초쯤 (2)번 과정의 웹툰 페이지 로딩이 끝났고, 21.9초쯤 새로고침 버튼을 눌렀다.)

위 사진을 보면, (2)번 과정과 같은 웹툰 페이지를 요청하였는데 이번 요청에 대해서는 DNS 패킷이 발생하지 않았다는 것을 알 수 있다. 그 이유는 OS의 DNS cache에서 ip 주소를 바로 가져오기 때문인데, (2)번 과정에서 comic.naver.com에 해당하는 ip주소를 매핑하면 OS는 DNS cache에 매핑 정보를 저장함으로써 이를 가능하게 한다. 다음의 간단한 콘솔 명령어로 DNS cache 데이터를 확인할 수 있다.

```
C:₩Users₩제호>ipconfig/displaydns > c:₩temp₩ipconfig.txt
```

ipconfig.txt - 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

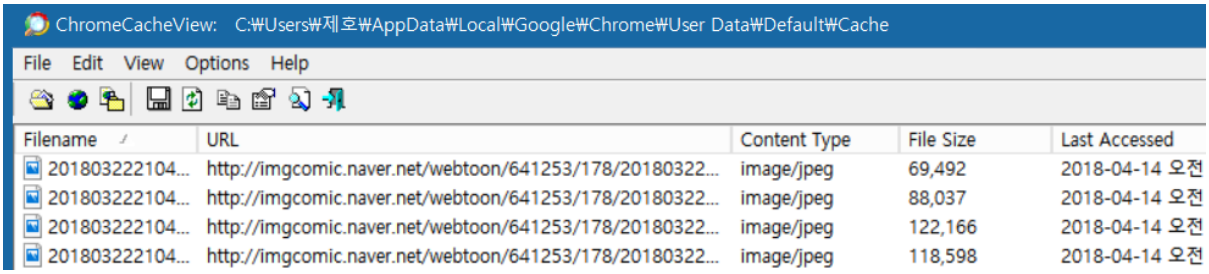
```
comic.naver.com
----------------------------------------
데이터 이름 . . . . . : comic.naver.com
데이터 유형 . . . . . : 5
TTL(Time To Live) . : 5
데이터 길이 . . . . . : 8
섹션 . . . . . : 응답
CNAME 레코드 . . . . : comic.naver.com.nheos.com

데이터 이름 . . . . . : comic.naver.com.nheos.com
데이터 유형 . . . . . : 5
TTL(Time To Live) . : 5
데이터 길이 . . . . . : 8
섹션 . . . . . : 응답
CNAME 레코드 . . . . : comic.nfront.nheos.com

데이터 이름 . . . . . : comic.nfront.nheos.com
데이터 유형 . . . . . : 1
TTL(Time To Live) . : 5
데이터 길이 . . . . . : 4
섹션 . . . . . : 응답
(호스트) 레코드 . . . : 117.52.137.140
```

위 명령어는 DNS cache에 저장된 정보를 지정한 경로에 ipconfig.txt라는 텍스트 파일로 저장하라는 명령어이다. 그 결과 ipconfig.txt 파일에는 comic.naver.com에 대한 매핑 정보가 입력되어 있고, 따라서 DNS Client는 comic.naver.com에 대한 ip 주소를 OS에서 바로 얻어왔음을 알 수 있다.

또한 (2)번 과정에서 웹툰 페이지를 처음 요청할 때 css, image 등의 요소들을 요청하여 수많은 HTTP 패킷이 발생하는 반면, 이번 과정에서 페이지를 다시 요청할 때는 약 20개 정도로 매우 적은 수의 HTTP 패킷만 발생했다(graph 참고). 그 이유를 검색해보니 크롬 브라우저가 가진 HTTP cache 기능 때문이라는 것을 알게 되었다. 크롬 브라우저의 HTTP cache 기능은 호스트 컴퓨터에 cache directory를 생성해 방문했던 페이지들의 콘텐츠 리소스를 일시적으로 저장해두고, 재방문 시 HTTP 요청을 할 필요 없이 cache 메모리로부터 바로 리소스를 가져올 수 있게 하는 기능이다. 크롬 브라우저 cache data 또한 확인할 수 있는데, 'ChromeCacheView' 라는 간단한 프로그램을 다운로드 받고 사용해봤다.



위 사진은 ChromeCacheView를 실행시킨 사진이다. 사진 위쪽의 디렉터리는 로컬 디스크의 크롬 cache 디렉터리를 나타내는데, 이 디렉터리에 저장된 cached resources의 목록을 보면 14일에 마지막으로 접근한 웹툰 페이지의 이미지 파일들임을 알 수 있다. 따라서 페이지를 재방문 했을 때, 브라우저가 웹툰 페이지의 리소스들을 HTTP cache에서 바로 로딩시켰음을 알 수 있다.

(4) DNS cache data 삭제 후 웹툰 페이지 재방문(새로고침) – packet No. 12458 ~

```
12458 32.6… 192.168.23.199   168.126.63.1   DNS      75 Standard query 0x5206 A comic.naver.com
12459 32.6… 168.126.63.1      192.168.23.199 DNS     272 Standard query response 0x5206 A comic.naver.com CNAME…
12471 35.2… 192.168.23.199    175.158.5.162  HTTP    558 GET /webtoon/detail.nhn?titleId=641253&no=178 HTTP/1.1
12498 35.3… 175.158.5.162     192.168.23.199 HTTP   1427 HTTP/1.1 200 OK  (text/html)
12499 35.3… 192.168.23.199    175.158.5.162  HTTP    556 GET /css/comic/common_20180410181119.css HTTP/1.1
12503 35.3… 192.168.23.199    175.158.5.162  HTTP    556 GET /css/comic/likeit_20180410181119.css HTTP/1.1
12511 35.3… 175.158.5.162     192.168.23.199 HTTP    517 HTTP/1.1 200 OK  (text/css)
12518 35.3… 175.158.5.162     192.168.23.199 HTTP    894 HTTP/1.1 200 OK  (text/css)
```

⋮

(이하 3-(2)과 같은 packet이 발생)

이번 과정을 통해 관찰하고 싶은 것은 DNS cache를 clear 시킨 후 재방문 했을 때 (3)번 과정과는 달리 DNS 메시지를 발생시킬 것이라는 사실이다. 이러한 예상은 틀리지 않았고, DNS 메시지가 나타났다.

그 과정을 살펴보면, 우선 (3)번 과정이 끝나고 `ipconfig/flushdns` 명령어를 통해 DNS cache를 비웠다. 그리고 크롬 설정에서 모든 인터넷 사용 기록을 지운 후, 웹툰 페이지를 재방문했다. 그 결과, 예상했듯이 (2)번 과정에서 나타난 DNS 메시지가 다시 발생했다. 다시 말해서, DNS cache에서 ip 주소를 얻을 수 없으므로 DNS 서비스를 요청했다.

그리고 추가적으로 (2)번 과정에서 발생한 많은 수의 HTTP 패킷이 똑같이 나타난 것을 확인했다(graph 참고). 이 사실을 통해 HTTP cache는 인터넷 사용 기록 삭제로 지워졌고 웹툰 페이지의 리소스를 다시 요청했음을 알 수 있다.

4. 새로 알게 된 사실

수업 때 배웠던 web cache나 DNS server cache외에도 호스트 컴퓨터의 OS와 웹 브라우저도 자체적으로 캐시 과정을 거친다는 사실을 잘 알게 되었다. 특히 캐시 목록을 직접 확인하고 비우기도 하며 캐시가 있는 경우와 없는 경우의 패킷 수를 비교할 수 있었고, 캐시가 네트워크 상의 메시지를 줄이는 역할을 잘 해내고 있다는 것을 실감할 수 있었다.

5. 관찰 후 소감

DNS 서비스는 client 관점에서 한 개의 query와 한 개의 response 메시지만 확인할 수 있기 때문에, HTTP처럼 서버에서 오는 많은 패킷을 확인 할 수 없다. 따라서 DNS로는 연속적인 traffic을 관찰 할 수 없었다. 또한 DNS Client에서 local DNS server로 전송된 DNS 메시지가 어떤 경로를 통해 이동하고 어떤 경로로 응답되는지 볼 수 없었다. 사실 wireshark 보고서 주제가 packet traffic 분석이기 때문에 이러한 경로 추적 문제는 큰 한계점이라고는 볼 수 없지만, 분산적으로 구현된 많은 DNS 서버들이 어떻게 상호작용하는지 패킷의 경로를 통해 알 수 없다는 것이 아쉬웠다. Client 입장에서 이러한 DNS 블랙박스 서비스는 유용하지만 보고서를 쓰는 입장에서는 아쉬움이 있었다.
처음에 wireshark를 써볼 때 어떤 기능들이 있는지 숙지하고 각 패킷이 의미하는 것이 무엇인지 파악하는데 어려움을 겪었다. 그 다음에 주제를 선정하는데 있어 많은 고민이 되었는데, 처음에는 FTP를 사용해 서버에 파일을 전송하고 HTTP로 해당 서버에 요청해 웹 페이지에 띄우는 과정으로 FTP, HTTP에 대해 보려고 했으나 서버 구축까지 가야 한다는 것을 알고 DNS로 전환했다. 처음에 말한 것처럼 DNS로 연속적인 traffic을 확인 할 수 없어 Wireshark를 사용해보는 보고서에 적합할지 의문이 들었는데, 직접 해보니 주제 선정을 잘 했다는 생각이 들었다. 40초의 웹 브라우징 만으로도 DNS 메시지의 구조와 mapping, caching 과정을 충분히 이해할 수 있었다. 또한 그 동안 네트워크의 계층 구조를 잘 이해하지 못했었는데, 이번 분석을 통해 패킷 정보와 해당 패킷에 포함된 protocol의 계층을 살펴보고 패킷의 발생 순서를 이해하면서 네트워크의 계층 구조에 대해 좀 더 알 수 있는 계기가 되어 좋았다.

6. 참고문헌

https://www.youtube.com/watch?v=72snZctFFtA (DNS를 설명하는 영상으로 실제 많은 도움이 되었다)
https://developers.google.com/web/fundamentals/performance/optimizing-content-efficiency/**http-caching**?hl=ko