CCOM 38

Dennis Bennhage, bennhage@student.chalmers.se
Hampus Lidin,   lidin@student.chalmers.se

HTTP)

1)

We can find the IP address of the host by looking in the 'Source' field in the HTTP GET message. Similarly, we obtain the server's address (gaia.cs.umass.edu) by looking in the 'Destination' field.

Source: 129.16.216.10
Destination: 128.119.245.12

2)

They are both using HTTP 1.1. This can be found in the first bits of the header.

3)

The status code is 200, which corresponds to the phrase 'Ok'.

4)

In the field 'Accept-language', you can see all the languages the browser accepts.

Accept-language: Swedish and English.

5)

This can be found in the 'Last-modified' field. However we noticed that this field wasn't updated after waiting several minutes and then refreshing the page. A clue to this is that maybe something was wrong on the server side, since the time was stuck at 6 o'clock (GMT) in the morning.

Last-modified: Wed, 06 May 2015 05:59:01 GMT.

6)

In the 'Content-length' field, it states the number of bytes in the body (the content that is actually being requested for).

Content-length: 128 bytes.

7)

No. Since the file wasn't in the cache, it needed to be downloaded from the server.

8)

Yes, under the 'Line-based text data' field.

9)

Yes, because the file hasn't been modified since the last time the file was downloaded.

If-Modified-Since: Wed, 06 May 2015 05:59:01 GMT.

10)

The status code 304 with the phrase "Not Modified" was returned by the server. As a consequence of this, the server has not explicitly included the contents of the file.

11)

It sent only one, which contains the request for the file.

12)

The next HTTP packet after the GET request was the response to the request. It had the status code 200 and phrase OK.

13)

4, including the response. The three first messages are 1460 bytes long (excluding the headers). The last on is 484 bytes long.

14)

4, one for the HTML file and three for the pictures.
Respectively, the addresses that the requests were directed for are:

128.119.245.12,
165.193.140.14 and
128.119.240.90 for the last two.

15)

They seem to be working in parallel since there is no way of telling if one transfer is done. TCP just keeps receiving messages from the servers and send responses in order to make the complete transfer flow as smooth as possible.

TCP)

1)

The source IP address can as before be found in the HTTP header, and the source port number is specified in the TCP header.

Source: 129.16.216.10.
Port: 57987.

2)

These values can be found in the same way as the question before.

Destination: 128.119.245.12.
Port: 80.

3)

3 segments were used; [SYN] from the host, [SYN, ACK] from the server and [ACK] from the host. These indicators can be found in the 'Flags' field.

4)

1, 1461, 2921, 4381, 5841 and 7301, respectively.

5)

1460 bytes.

6)

Post (1st seg.):	6.792884s.
2nd seg.:	6.792895s.
1st ack.:	6.914991s.
3rd seg.:	6.915070s.
4th seg.:	6.915083s.
5th seg.:	6.915093s.
6th seq.:	6.915103s.
2nd ack.:	7.040115s.

7)

Please refer to the appendix for the graph of the TCP session.

Segment	RTT (ms)	Estimated RTT (ms)
1	122.107	122.107
2	122.096	0.875*122.107 + 0.125*122.096 = 122.106
3	125.045	0.875*122.106 + 0.125*125.045 = 122,473
4	125.032	0.875*122.473 + 0.125*125.032 = 122,793
5	125.022	0.875*122.793 + 0.125*125.022 = 123,072
6	125.012	0.875*123.072 + 0.125*125.012 = 123,314

8)

The next byte that should be sent by the host and acknowledged by the server. The server determines this number by assuring that each prior byte has reached the server and then includes the next byte's number in the Acknowledgment number field.

9)

Around 300 bytes after a while. It started with a narrower window and then increased the window as more packets were sent. Later on the window size was 600-700 bytes, until transfer completion.

10)

All the time. The first ACK message acknowledged the first two segments, and then roughly between 4-10 segments are acknowledged at the same time.

11)

Round	#Segments
1	2
2	4
3	8
4	12
5	22
6	34
7	60
8	56
9	13

The slow-start phase is identified by the increase of the number of segments sent in a window. It starts in round one, with a safe window of two segments. Then, after the first ACK has been received, it can calculate the RTT and expand the window if possible. This phase goes on until congestion avoidance takes over, which the name indicates avoids congestion. This takes place in round 8, where the number of segments sent in a window is more or less the same. The last round is the remaining bytes in the file.

12)

The number of bytes that the server accepts can be found in the 'Window size value' field in the TCP header. This is increased as the receiver (the server) realizes that it can take more bytes at the same time. When the buffer starts to get full, this window starts stabilizing at a decent value.

13)

We assume that the communication began at the first handshake message and ended at the last TCP segment, which are at 6.668835s and 8.098057s, respectively. That yields a total transfer time of 1.429222s. We get the size of the complete transfer by filtering the packets in Wireshark to only show the TCP packets sent from the host computer, which in this case was 316 595*8 = 2 532 760 bits. Then we just divide the total number of bits by the total elapsed time and get 1,772 MBit/s.

DNS)

1)

We will ask about the "Value", which in this case is the IP address of the host name www.webafrica.co.za:

```
> nslookup www.webafrica.co.za

Server:  res1.chalmers.se
Address:  129.16.1.53

Non-authoritative answer:
Name:    www.webafrica.co.za
Address:  41.185.61.34
```

2)

Here we will ask about the name servers that knows about the IP address of the domain ufrj.br:

```
> nslookup -type=ns ufrj.br

Server:  res1.chalmers.se
Address:  129.16.1.53

Non-authoritative answer:
ufrj.br nameserver = ns4.ufrj.br
ufrj.br nameserver = ns.ufrj.br
ufrj.br nameserver = ns2.ufrj.br
ufrj.br nameserver = ns3.ufrj.br
```

```
ns.ufrj.br      internet address = 146.164.170.11
ns2.ufrj.br     internet address = 200.20.116.90
ns3.ufrj.br     internet address = 146.164.150.11
ns4.ufrj.br     internet address = 200.156.137.11
```

3)

Here we ask about the canonical name for the mail server of the domain amazon.com, by sending the request through name server ns2.ufrj.br:

```
> nslookup -type=mx amazon.com ns2.ufrj.br

Server:  stic-dns2.tic.ufrj.br
Address:  200.20.116.90

*** stic-dns2.tic.ufrj.br can't find amazon.com: Query refused
```

We didn't get an answer since ns2.ufrj.br is a dedicated name server for the domain ufrj.br.

4)

They both use UDP. The reason UDP is used instead of TCP is because no data is being sent, thus the need for a reliable and connected transfer is redundant. Also UDP is a lot faster.

5)

Port 53.

6)

Also port 53.

7)

To IP address 129.16.1.53, which both are found in the IP message and by running 'ipconfig /all' command.

8)

It's a standard query. Some sections that are included in the query are 'Flags' field, transaction ID, number of questions, 'Queries' field. Here no RR:s are provided.

9)

Same as in the query message, with the addition of 'Answers', 'Authoritative nameservers' and 'Additional records' fields. It contains RR:s for the query, one of them being the canonical name (CNAME) for the hostname and the other two being the addresses for two servers. In addition the answers also provide TTL.

10)

No, every image is sent after the initial DNS request. Since the IP address is known, we don't need to query for it each time we fetch an image from that server.

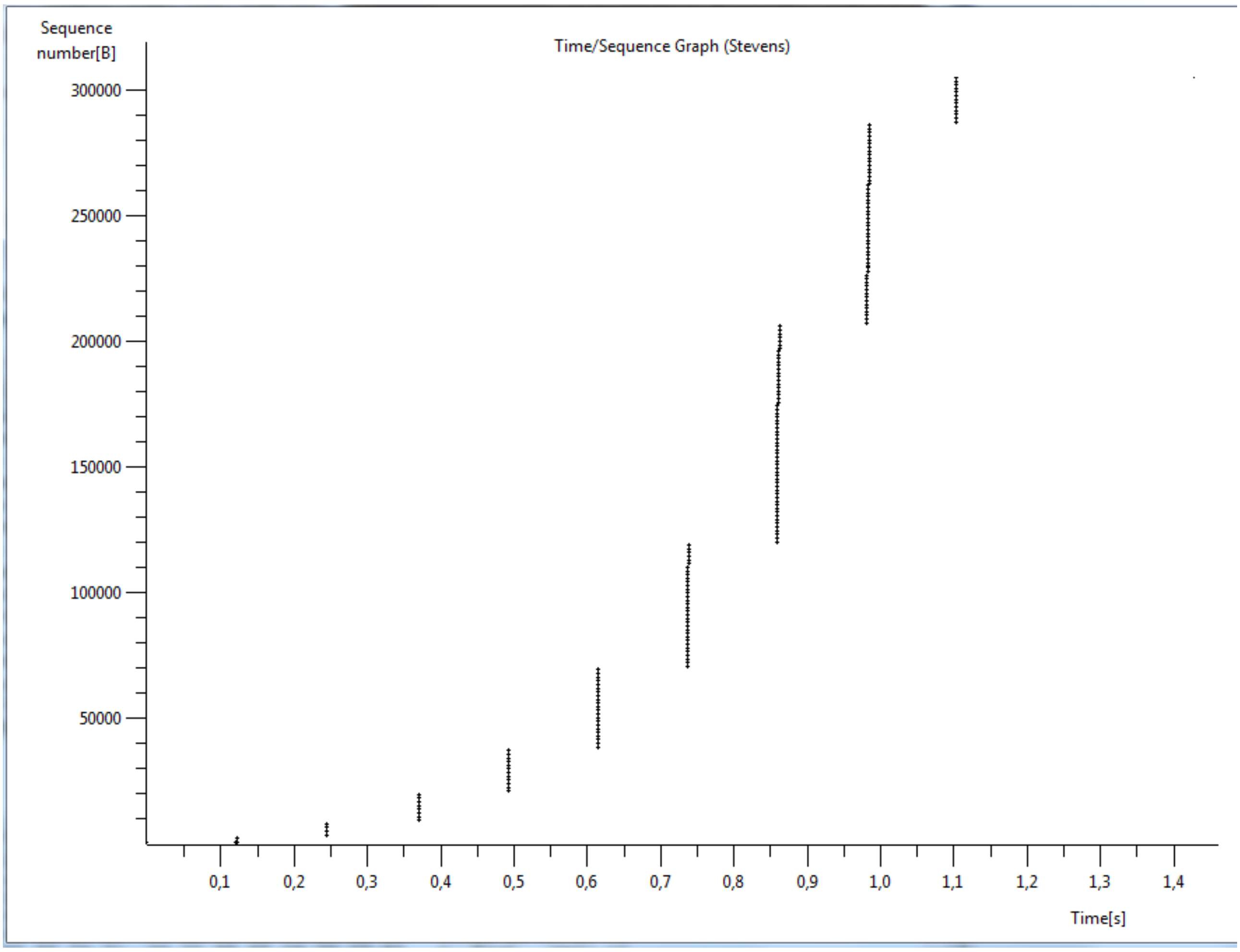
11)

The 'Flags' contain the several bits that indicate the status of the message. The 'Queries' contain the requests from the host. The 'Answers' contain the results from the DNS server. 'Authoritative nameservers' contain the name servers of the requested hostname. 'Additional records' contain the IP addresses for the name servers.

12)

The first RR is the 'Name' for the server; it's meaning depends on the 'Type' of server. That's what the second field is, which in this case it is Type A ('Name' is then a hostname). The third field is the servers 'Class' (IN (0x0001)). The fourth field is the 'TTL' value for the server's IP address. The fifth field indicates the data length of the IP address, in this case 4 bytes (IPv4). The last field states the IP address of the server.

Appendix


Sequence number[B]
Time/Sequence Graph (Stevens)
300000
250000
200000
150000
100000
50000
0,1
0,2
0,3
0,4
0,5
0,6
0,7
0,8
0,9
1,0
1,1
1,2
1,3
1,4
Time[s]