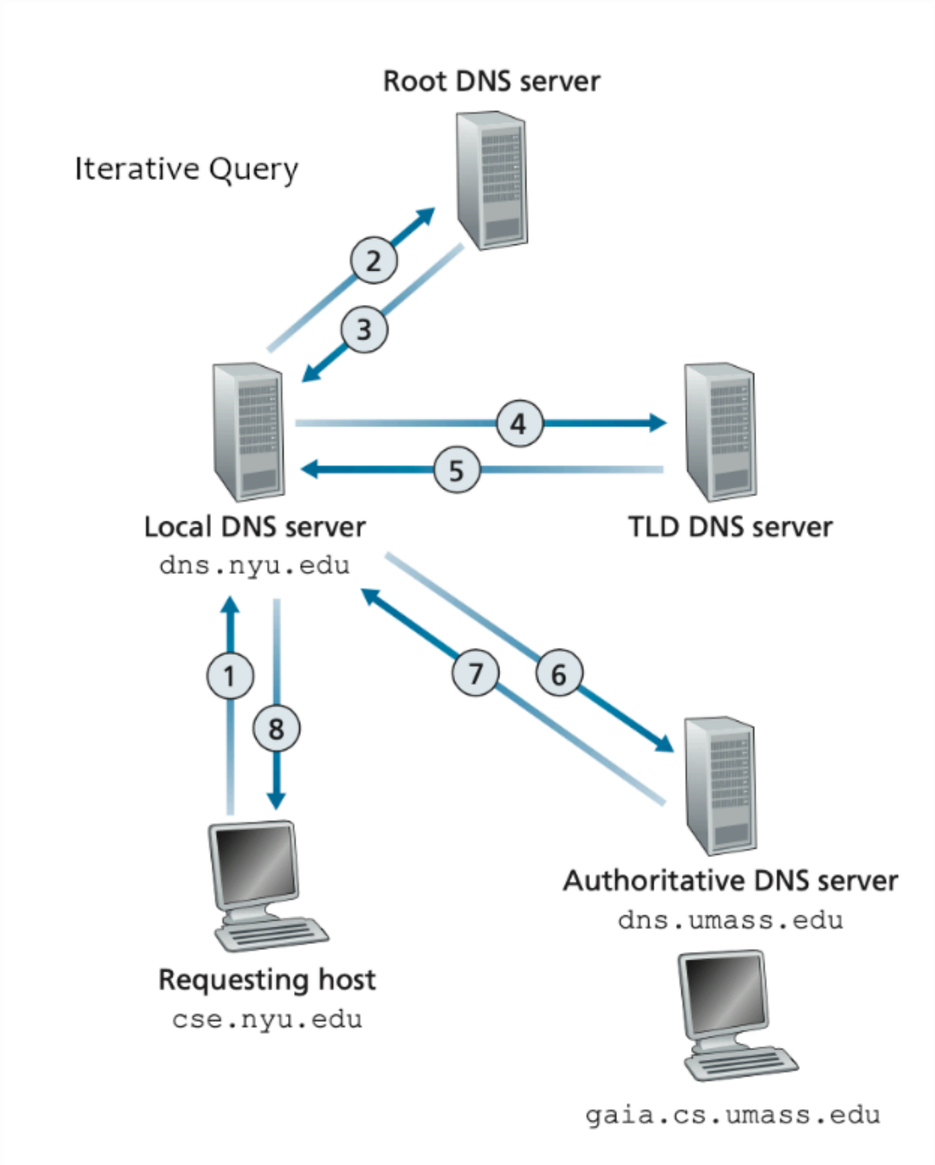
Computer Networks

Malik Algazaeery

Lab-3

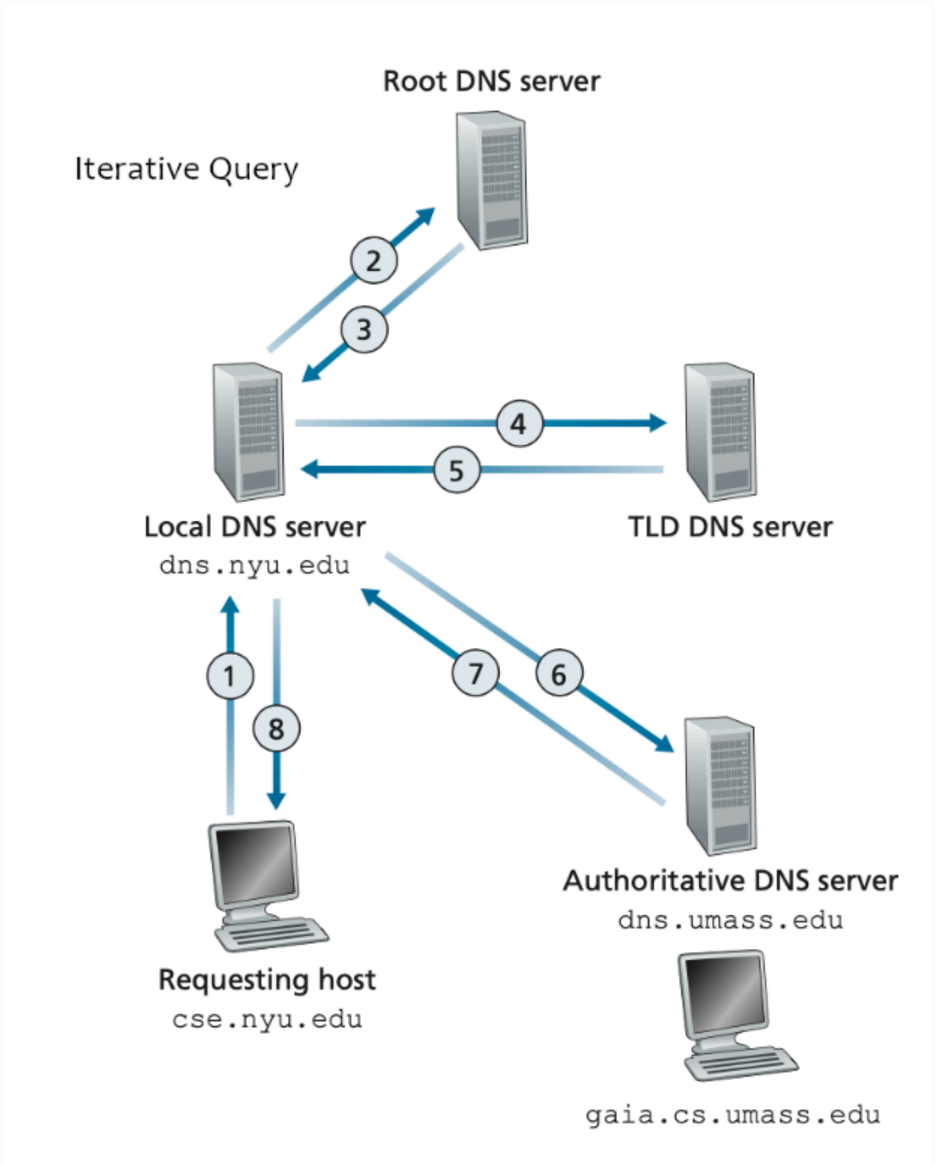
DNS - Iterative vs Recursive Query

1. Between steps 1 and 2, where does the Local DNS server check first? Answer with 'User', 'DNS Local', 'DNS Root', 'DNS TLD', or 'DNS Authoritative'.
2. Between steps 2 and 3, assuming the root DNS server doesn't have the IP we want, where does the response link? Answer with 'DNS Local', 'DNS Root', 'DNS TLD', or 'DNS Authoritative'.
3. Between steps 4 and 5, assuming the TLD DNS server doesn't have the IP we want, where does the response link? Answer with 'DNS Local', 'DNS Root', 'DNS TLD', or 'DNS Authoritative'.
4. Which type of query is considered best practice: iterative or recursive?

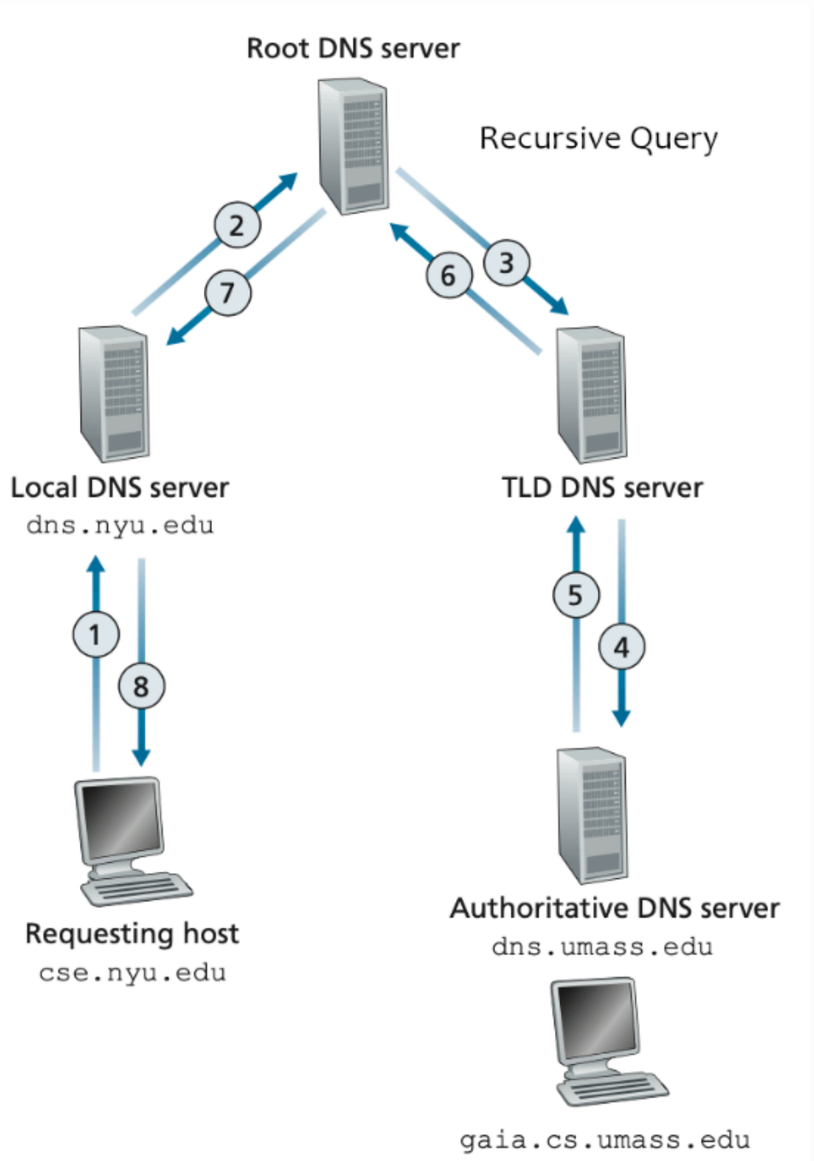


DNS - Iterative vs Recursive Query

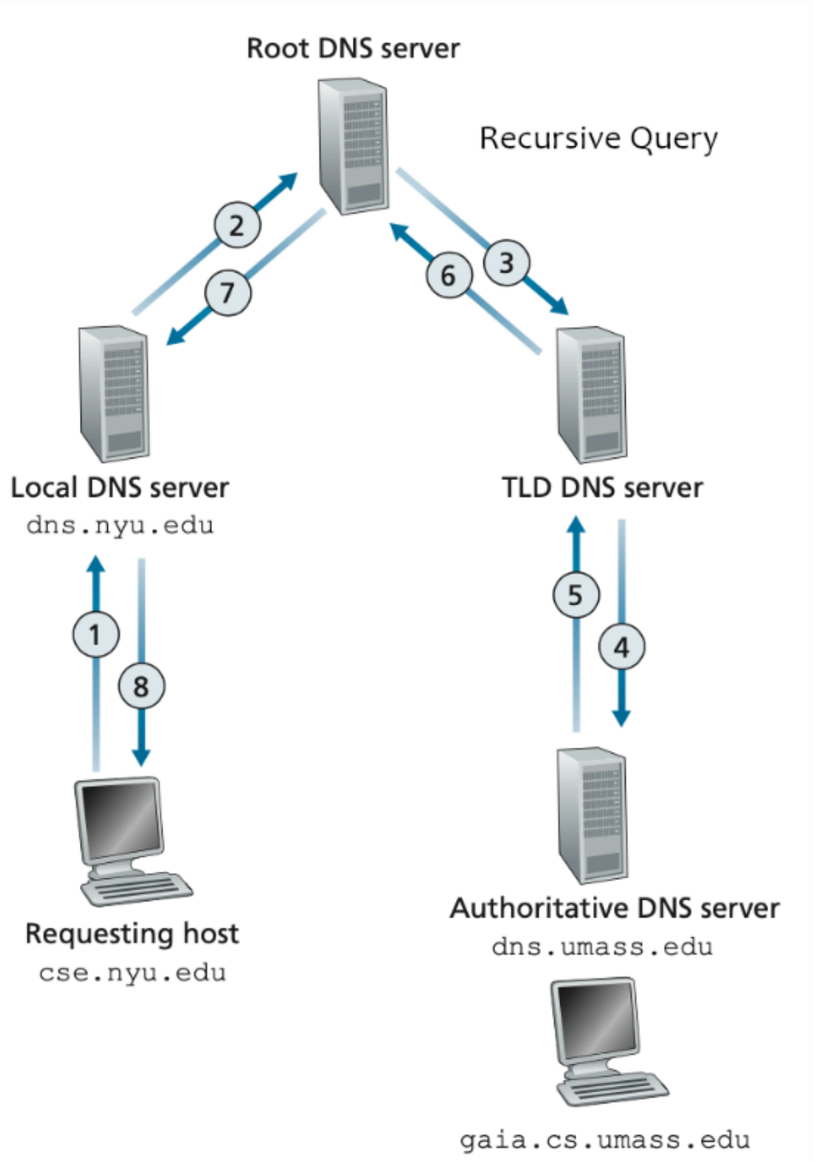
1. Between steps 1 and 2, where does the Local DNS server check first? Answer with 'User', 'DNS Local', 'DNS Root', 'DNS TLD', or 'DNS Authoritative'.
    - DNS Root
2. Between steps 2 and 3, assuming the root DNS server doesn't have the IP we want, where does the response link? Answer with 'DNS Local', 'DNS Root', 'DNS TLD', or 'DNS Authoritative'.
    - DNS TLD.
3. Between steps 4 and 5, assuming the TLD DNS server doesn't have the IP we want, where does the response link? Answer with 'DNS Local', 'DNS Root', 'DNS TLD', or 'DNS Authoritative'.
    - DNS Authoritative.
4. Which type of query is considered best practice: iterative or recursive?
    - Iterative



1. Between steps 1 and 2, where does the Local DNS server check first? Answer with 'User', 'DNS Local', 'DNS Root', 'DNS TLD', or 'DNS Authoritative'.
2. Between steps 2 and 3, where does the root DNS forward the request to? Answer with 'DNS Local', 'DNS Root', 'DNS TLD', or 'DNS Authoritative'.
3. Between steps 4 and 5, where does the authoritative DNS forward the response to? Answer with 'DNS Local', 'DNS Root', 'DNS TLD', or 'DNS Authoritative'.

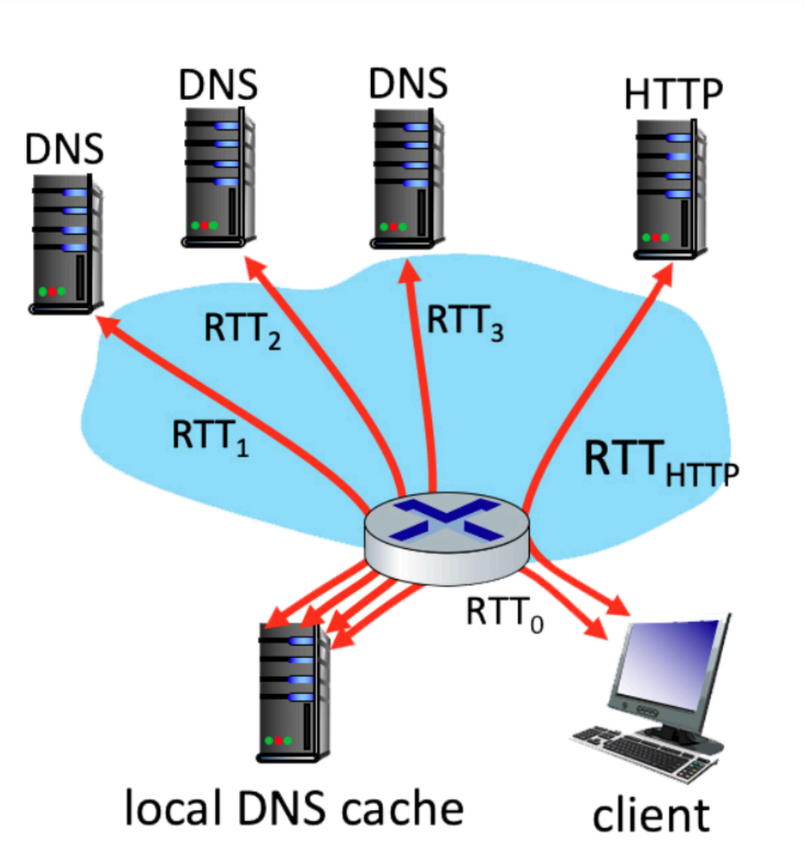


1. Between steps 1 and 2, where does the Local DNS server check first? Answer with 'User', 'DNS Local', 'DNS Root', 'DNS TLD', or 'DNS Authoritative'.
    - DNS Root.
2. Between steps 2 and 3, where does the root DNS forward the request to? Answer with 'DNS Local', 'DNS Root', 'DNS TLD', or 'DNS Authoritative'.
    - DNS TLD
3. Between steps 4 and 5, where does the authoritative DNS forward the response to? Answer with 'DNS Local', 'DNS Root', 'DNS TLD', or 'DNS Authoritative'.
    - DNS TLD.



DNS and HTTP delays

Suppose within your Web browser you click on a link to obtain a Web page. The IP address for the associated URL is not cached in your local host, so a DNS lookup is necessary to obtain the IP address. Suppose that four DNS servers are visited before your host receives the IP address from DNS. The first DNS server visited is the local DNS cache, with an RTT delay of RTT_0 = 4 msecs. The second, third and fourth DNS servers contacted have RTTs of 2, 36, and 6 msecs, respectively. Initially, let's suppose that the Web page associated with the link contains exactly one object, consisting of a small amount of HTML text. Suppose the RTT between the local host and the Web server containing the object is RTT_{HTTP} = 24 msecs.



1. Assuming zero transmission time for the HTML object, how much time (in msec) elapses from when the client clicks on the link until the client receives the object?
2. Now suppose the HTML object references 4 very small objects on the same server. Neglecting transmission times, how much time (in msec) elapses from when the client clicks on the link until the base object and all 4 additional objects are received from web server at the client, assuming non-persistent HTTP and no parallel TCP connections?
3. Suppose the HTML object references 4 very small objects on the same server, but assume that the client is configured to support a maximum of 5 parallel TCP connections, with non-persistent HTTP.
4. Suppose the HTML object references 4 very small objects on the same server, but assume that the client is configured to support a maximum of 5 parallel TCP connections, with persistent HTTP.
5. What's the fastest method we've explored: Nonpersistent-serial, Nonpersistent-parallel, or Persistent-parallel?

1. The time from when the Web request is made in the browser until the page is displayed in the browser is: $RTT_0 + RTT_1 + RTT_2 + RTT_3 + 2*RTT_{HTTP} = 4 + 2 + 36 + 6 + 2*24 = 96$ msecs. Note that 2 RTT_{HTTP} are needed to fetch the HTML object - one RTT_{HTTP} to establish the TCP connection, and then one RTT_{HTTP} to perform the HTTP GET/response over that TCP connection.

2. The time from when the Web request is made in the browser until the page is displayed in the browser is: $RTT_0 + RTT_1 + RTT_2 + RTT_3 + 2*RTT_{HTTP} + 2*4*RTT_{HTTP} = 4 + 2 + 36 + 6 + 2*24 + 2*4*24 = 288$ msecs. Note that two RTT_{HTTP} delays are needed to fetch the base HTML object - one RTT_{HTTP} to establish the TCP connection, and one RTT_{HTTP} to send the HTTP request, and receive the HTTP reply. Then, serially, for each of the 4 embedded objects, a delay of $2*RTT_{HTTP}$ is needed - one RTT_{HTTP} to establish the TCP connection and then one RTT_{HTTP} to perform the HTTP GET/response over that TCP connection.

3. Since there are only 4 objects, there's a delay of 48 msec for the DNS query, two RTT_{HTTP} for the base page, and $2*RTT_{HTTP}$ for the objects since the requests for these can be run in parallel. The total is 48 + 48 + 48 = 144 msec. As in 2 above, 2 RTT_{HTTP} are needed to fetch the base HTML object - one RTT_{HTTP} to establish the TCP connection, and one RTT_{HTTP} to send the HTTP request and receive the HTTP reply containing the base HTML object. Once the base object is received at the client, the 4 HTTP GETS for the embedded objects can proceed in parallel. Each (in parallel) requires two RTT_{HTTP} delays - one RTT_{HTTP} to set up the TCP connection, and one RTT_{HTTP} to perform the HTTP GET/response for an embedded object.

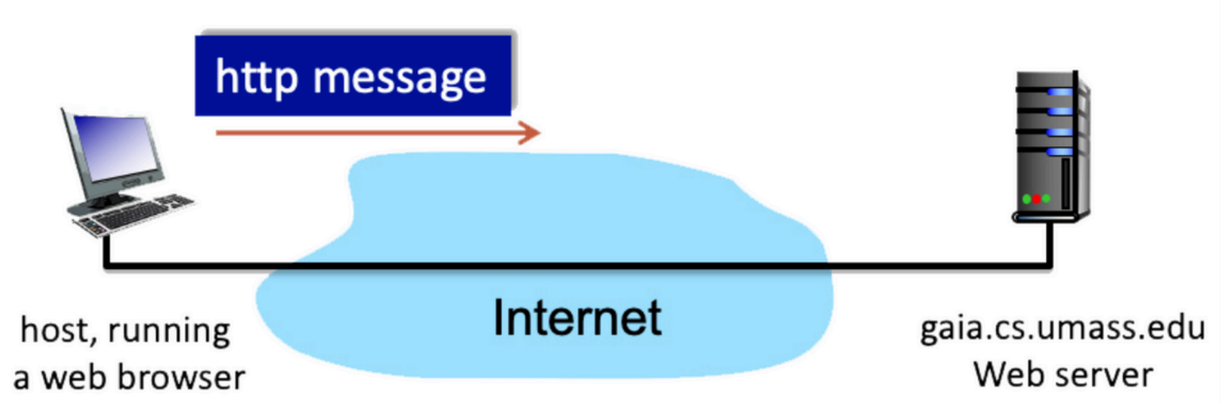
4. Since there are only 4 objects, there's a delay of 48 msec for the DNS query, two RTT_{HTTP} for the base page, and 1 RTT_{HTTP} for the objects. The total is 48 + 48 + 24 = 120 msec.As in 2 and 3 above, two RTT_{HTTP} delays are needed to fetch the base HTML object - one RTT_{HTTP} to establish the TCP connection, and one RTT_{HTTP} to send the HTTP request, and receive the HTTP reply containing the base HTML object. However, with persistent HTTP, this TCP connection will remain open for future HTTP requests, which will therefore not incur a TCP establishment delay. Once the base object is received at the client, the maximum of five requests can proceed in parallel, each retrieving one of the 4 embedded objects. Each (in parallel) requires only one RTT_{HTTP} delay to perform the HTTP GET/response for an embedded object. Once these first five objects have been retrieved, (if necessary) the remaining embedded objects can be retrieved (in parallel). This second round of HTTP GET/response to retreive the remaining embedded objects takes only one more RTT_{HTTP}, since the TCP connection has remained open.

5. The delay when using persistent parallel connections is faster than using nonpersistent parallel connections, which is faster than using nonpersistent serial connections.

The HTTP GET message

Consider the figure below, where a client is sending an HTTP GET message to a web server, gaia.cs.umass.edu. Suppose the client-to-server HTTP GET message is as below:

1. What is the name of the file that is being retrieved in this GET message?
2. What version of HTTP is the client running?
3. True or False: The client will accept html files
4. True or False: The client will accept jpeg images
5. What is the client's preferred version of English?
6. What is the client's least preferred version of English?
7. True or False: The client will accept the German language
8. True or False: The client already has a cached copy of the file



GET /kurose_ross_sandbox/interactive/quotation3.htm HTTP/1.1
Host: gaia.cs.umass.edu
Accept: text/plain, text/html, text/xml, image/gif, image/jpeg, audio/vnf.wave, audio/mpeg, video/mp4, video/wmv,
Accept-Language: en-us, en-gb;q=0.4, en;q=0.9, fr, fr-ch, fi, ar
If-Modified-Since: Tue, 13 May 2025 02:11:23 -0700
User Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_7_3) AppleWebKit/534.53.11 (KHTML, like Gecko) Version/5.1.3 Safari/534.53.10

1. The name of the file is quotation3.htm.

2. The client is running on HTTP/1.1

3. True. In the 'Accept' field the client includes 'text/html' files.

4. True. The client does include 'image/jpeg' in its 'Accept' field.

5. The client's preferred version of English is American English. Any language without a defined q value has a default value of 1

6. The client's least preferred version of English is British English because it has the lowest q value.

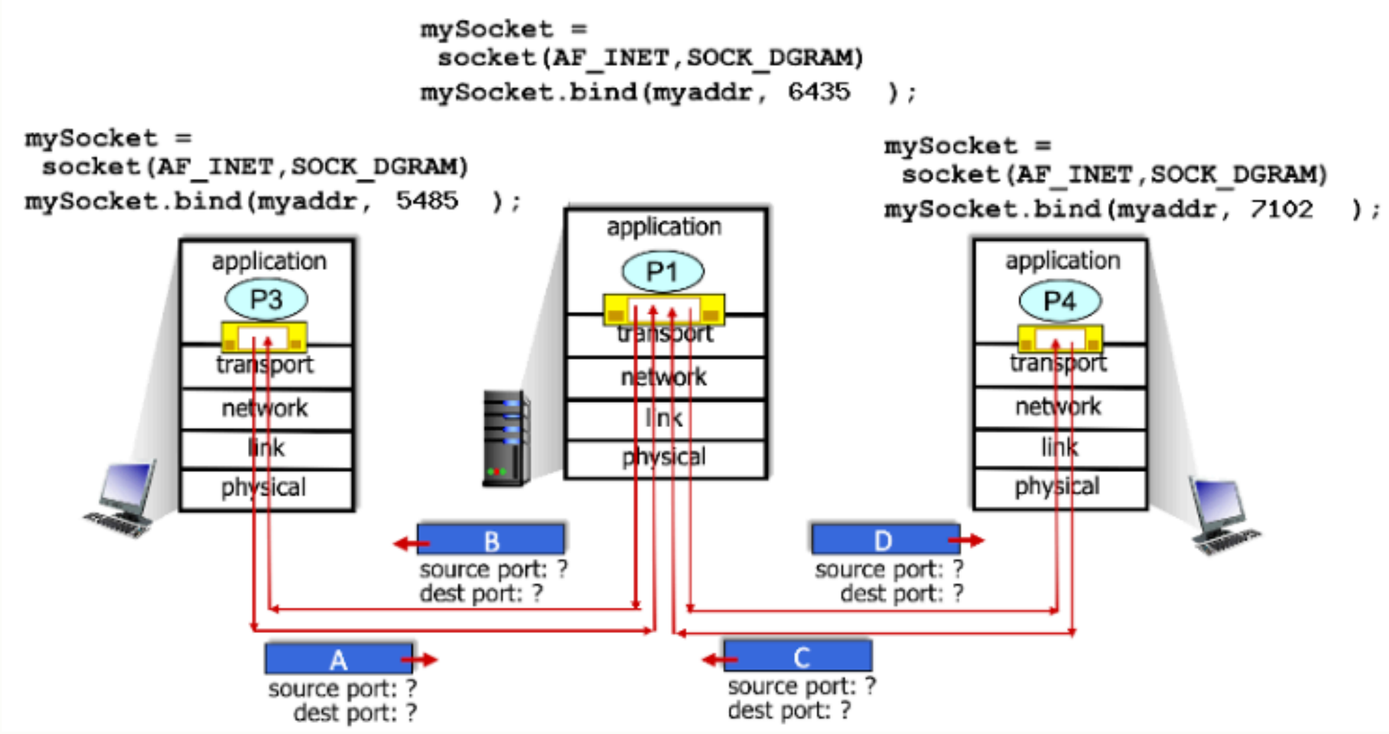
7. False. The client does NOT include German in its 'Accepted-Language' field.

8. True. The client has a cached copy of the file that was updated on: Tue, 13 May 2025 02:11:23 -0700

UDP Multiplexing and Demultiplexing

In the scenario below, the left and right clients communicate with a server using UDP sockets. The same socket at the server is used to communicate with both clients. The Python code used to create the sockets is shown in the figure. Consider the four transport-layer packets – A, B, C and D – shown in the figure.

1. What is the source port # for packet D?
2. What is the destination port # for packet D?
3. What is the source port # for packet A?
4. What is the destination port # for packet A?
5. What is the source port # for packet C?
6. What is the destination port # for packet C?
7. What is the source port # for packet B?
8. What is the destination port # for packet B?



UDP Multiplexing and Demultiplexing

In the scenario below, the left and right clients communicate with a server using UDP sockets. The same socket at the server is used to communicate with both clients. The Python code used to create the sockets is shown in the figure. Consider the four transport-layer packets – A, B, C and D – shown in the figure.

1. What is the source port # for packet D?
    - 6435.
2. What is the destination port # for packet D?
    - 7102.
3. What is the source port # for packet A?
    - 5485.
4. What is the destination port # for packet A?
    - 6435.
5. What is the source port # for packet C?
    - 7102.
6. What is the destination port # for packet C?
    - 6435.
7. What is the source port # for packet B?
    - 6435.
8. What is the destination port # for packet B?
    - 5485.