Computer Networks Homework 2

Reference Solution

1. (20%) Consider a circular Distributed Hash Table (DHT) with node identifiers in the range [0; 15]. Suppose there are seven peers with identifiers 1, 3, 4, 5, 8, 12 and 14. Suppose that peer 3 learns that peer 5 has left the DHT. How does peer 3 update its successor state information? Which peer is now its first successor? Its second successor?

Ans: Peer 3 learns that peer 5 has just left the system, so peer 3 asks its first successor (peer 4) for the identifier of its immediate successor (peer 8). Then peer 3 will make peer 8 as its second successor. Note: Peer 3 knows that peer 5 was originally the first successor of peer 4, so peer 3 would wait until peer 4 finishes updating its first successor.

2. (20%) Suppose within your Web browser you click on a link to obtain a Web page. Suppose that the IP address for the associated URL is not cached in your local host, so that a DNS lookup is necessary to obtain the IP address. Suppose that n DNS servers are visited before your host receives the IP address from DNS; the successive visits incur an RTT of $RTT_1$, ..., $RTT_n$. Further suppose that the Web page associated with the link contains exactly one object, a small amount of HTML text. Let $RTT_0$ denote the RTT between the local host and the server containing the object. Assuming the transmission time of the object is zero. Suppose the HTML file indexes three very small objects on the same server. Neglecting transmission times, how much time elapses with

(a) nonpersistent HTTP with no parallel TCP connections,
(b) nonpersistent HTTP with parallel connections,
(c) persistent HTTP with pipelining?

Ans:
3. (20%) Suppose you wanted to do a transaction from a remote client to a server as fast as possible. Would you use UDP or TCP? Why?

Ans: You would use UDP. With UDP, the transaction can be completed in one roundtrip time (RTT) - the client sends the transaction request into a UDP socket, and the server sends the reply back to the client's UDP socket. With TCP, a minimum of two RTTs are needed - one to set-up the TCP connection, and another for the client to send the request, and for the server to send back the reply.

4. (20%) Consider the figure shown below, for which there is an institutional network connected to the Internet. Suppose that the average object size is 900,000 bits and that the average request rate from the institution’s browsers to the origin servers is 15 requests per second. Also suppose that the amount of time it takes from when the router on the Internet side of the access link forwards an HTTP request until it receives the response is two seconds on average. Model the total average response time as the sum of the average access delay (that is, the delay from Internet router to institution router) and the average Internet delay. For thee average access delay, use $\Delta/(1-\beta)$, where $\Delta$ is the average time required to send an object
over the access link and $\beta$ is the arrival rate of objects to the access link.

(a) Find the total average response time.
(b) Now suppose a cache is installed in the institutional LAN. Suppose the hit rate is 0.4. Find the total response time.

Ans:

(a)  
\[
\Delta = \frac{900000 \text{ bits}}{15000000 \text{ bps}} = 0.06 \text{ sec}
\]
\[
\Delta \beta = 0.06 \text{ sec/request} \times 15 \text{ request/sec} = 0.9
\]
\[
\Delta / (1 - \Delta \beta) = 0.06 \text{ sec} / (1 - 0.9) = 0.6
\]

Total average response time = 0.6 + 2 = 2.6 sec

(b)  
The traffic intensity on the access link is reduced by 40% since the 40% of the requests are satisfied within the institutional network. Thus the average access delay is 0.06/[1 - 0.6 * 0.9] = 0.1304 sec. The response time is approximately zero if the request is satisfied by
the cache (which happens with probability 0.4); the average response time is 0.1304 sec + 2 sec = 2.1304 sec for cache misses (which happens 60% of the time). So the average response time is 0.4 * 0 sec + 0.6 * 2.1304 = 1.27824 sec. Thus the average response time is reduced from 2.6 sec to 1.27824 sec.

5. (20%) Two HTTP request methods are GET and POST. Are there any other methods in HTTP/1.0? If so, what are they used for? Are there other methods in HTTP/1.1?

Ans:

(a)

**HEAD**: Asks for the response identical to the one that would correspond to a GET request, but without the response body. This is useful for retrieving meta-information written in response headers, without having to transport the entire content.

(b)

**PUT**: Uploads a representation of the specified resource.

**DELETE**: Deletes the specified resource.

**TRACE**: Echoes back the received request, so that a client can see what (if any) changes or additions have been made by intermediate servers.

**OPTIONS**: Returns the HTTP methods that the server supports for specified URL. This can be used to check the functionality of a web server by requesting instead of a specific resource.

**CONNECT**: Converts the request connection to a transparent TCP/IP tunnel, usually to facilitate SSL-encrypted communication (HTTPS) through an unencrypted HTTP proxy.