Assignment #01

1. In this problem, we compare non-persistent HTTP with persistent HTTP. Suppose the page that your browser wants to download is 300K bits long, and contains 5 embedded images, each of which is 200K bits in length. The page and the five images are all stored on the same server, which has a 300ms RTT from your browser. We will abstract the network path between your browser and the web server as a 100Mbps "link". You can assume that the time it takes to transmit a GET message into the "link" is zero, but you should account for the time it takes to transmit the base page and the embedded objects into the "link". In your answer below, make sure to take into account the time needed to setup up TCP connections (1 RTT). Note: Please give the calculation process in detail. (Don't consider the time needed to close TCP connections)

(1) Assume non-persistent HTTP (and assuming no parallel connections are open between the browser and server). How long is the **response time**, i.e., from the time when the user requests the URL to the time when the page and its embedded objects are displayed?

(2) Now assume persistent HTTP. What is the **response time**, assuming no pipeline?

(3) Now assume persistent HTTP, but assume that the browser can use pipeline. What is the **response time**?

Answer:

(1).To transmit a page and 5 embedded images,the web server and browser need to build 6 times connection.The total response time of each connection consists of 2 RTT(①the first two parts of three_times_handshake ②the third handshake with a HTTP request and the web server's response) and the time at which the server transfers the file.

Time: $2*RTT+\frac{300\text{kb}}{v}+(2*RTT+\frac{200kb}{v})*5$

So the response time,: $2*300\text{ms}*6+\frac{300k+200k*5}{100Mbps}=3613ms$

(2)When no pipeline: the web server and the browser build one time TCP persistent connection.The total time include one TCP connection(1 RTT) and 6 request/response time(6 RTT) and file transfer time.

Time: $RTT+6*RTT+\frac{300\text{k}+200k*5}{v}$

So the response time: $7*300ms+\frac{300k+200k*5}{100Mbps}=2113\text{ms}$

(3) Use pipeline:

First ,the browser and the web server need to build a TCP persistent connection.

the first two handshakes between the client and server occupy an RTT.

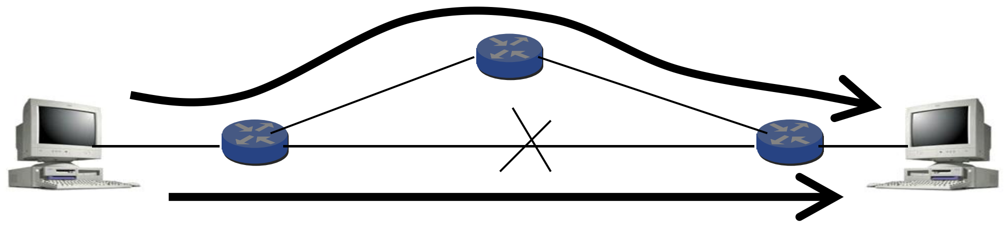
Second, the client and server build the third handshake and send the request to the server at the same time.Client receive the response .This occupy an RTT +request transfer time.

Third ,the client find 5 embedded images and send all request at one time and receive response from the server.This occupy an RTT+5 images transfer time.

Time: $$RTT+(RTT+\frac{300K}{V})+(RTT+\frac{200k*5}{v})$$

The response time: $$3*300ms+\frac{300k+200k*5}{100Mbps}=913ms$$

2. Suppose two hosts have a long-lived TCP session over a path with a 100ms round-trip time (RTT). Then, a link fails, causing the traffic to flow over a longer path with a 500ms RTT.



(1) Suppose the router on the left recognizes the failure immediately and starts forwarding data packets over the new path, without losing any packets. (Assume also that the router on the right recognizes the failure immediately and starts directing ACKs over the new path, without losing any ACK packets.) Why might the TCP sender retransmit some of the data packets anyway?

(2) Suppose instead that the routers do not switch to the new paths all that quickly, and the data packets (and ACK packets) in flight are all lost. What new congestion window size does the TCP sender use? Why?

Answer:

(1).The TCP makes its RTO based on Esltimated RTT,RTT≈100ms,RTO double≈200ms.Although the two routers have recognized the failure ,make the correct judgment and changed the path.The sender didn’t change the timeout value,and this transfer spends a long time-500ms which has far exceeded the retransmission timeout 200ms.So when reach timeout,the sender didn’t receive the ACK from the receiver.It makes the sender retransmit some packets anyway.

(2)

New congestion window size: cwnd=1.

The cwnd is changed by how the packet losses were detected.When the receiver receive 3 redundancy ACK ConWin will be half.The data packets and the ACK was lost and the timeout was not be changed when the links fail.So when reach the timeout,the sender determined the network congested and makes the cwnd=1 begin Slow Start by TCP Congestion Control.