Part 4:

Single process server:

Below are the readings for requesting and receiving of 1 to 10 files on persistent and non-persistent connections.

Single process server		
No of files requested	Persistent	Non-persistent
1	0.107773	0.108704
2	0.200963	0.205762
3	0.285675	0.288256
4	0.377794	0.380607
5	0.465674	0.467949
6	0.559913	0.556248
7	0.663044	0.66858
8	0.789674	0.759381
9	0.84718	0.870176
10	0.939196	0.982681

As we can see, it takes a little bit more time for non-persistent connections as compared to persistent connection. This difference in the time to serve the same number of files on two different connections is because of the time to close and establish a new connection during non-persistent connections.

Trends:

When two requests are fired simultaneously, follow cases are happening.

1. 1st Fire → Persistent → This will fire 10 persistent http requests to get 10 files
   2nd Fire → Persistent → This will fire 10 persistent http requests to get 10 files

   In this case, second request will not be served until the first request is served entirely.

2. 1st Fire → Persistent → This will fire 10 persistent http requests to get 10 files
   2nd Fire → Non-persistent → This will fire 10 non-persistent http requests to get 10 files

   In this case, second request will not be served until the first request is served entirely.

3. 1st Fire → Non-persistent → This will fire 10 non-persistent http requests to get 10 files
   2nd Fire → Persistent → This will fire 10 persistent http requests to get 10 files

   In this case, when the 1st fire closes connection after serving 1st http request, server might start serving 2nd fire and as it is a persistent fire, first fire will not be served until all the persistent http requests by 2nd fire are served.

4. 1st Fire → Non-persistent → This will fire 10 non-persistent http requests to get 10 files
   2nd Fire → Non-persistent → This will fire 10 non-persistent persistent http requests to get 10 files

   In this case, when the 1st fire closes connection after serving 1st http request, server might start serving non-persistent http request by 2nd fire. Once a that is served, server might server another http request by any of the fires and thus, serving both the fires by switching between as both the fires are non-persistent.

Multithreaded server:

Below are the readings for requesting and receiving of 1 to 10 files when two requests are fired simultaneously from different clients for both persistent and non-persistent connections.

Multithreaded Server				
Number of files request in each request	Persistent		Non-Persistent	Non-persistent
	Request-1	Request-2	Request-1	Request-2
1	0.111063	0.137196	0.137643	0.131841
2	0.245483	0.250379	0.255714	0.247146
3	0.358474	0.341432	0.39902	0.348339
4	0.438713	0.476341	0.444554	0.475436
5	0.573883	0.530752	0.539006	0.53252
6	0.653179	0.64984	0.658949	0.673186
7	0.733164	0.780739	0.763757	0.761638
8	0.826864	0.843068	0.860968	0.877887
9	0.918173	0.95805	1.00505	0.971665
10	1.08575	1.05615	1.09681	1.07079

In single process server, if two requests are fired simultaneously from two different clients/terminals, second request is not served until the first request is served by the server and it depends whether the fired requests are persistent or non-persistent.

Whereas, in case of multithreaded server, when two requests are fires from two different clients, both the requests are served simultaneously taking approximately same time as a separate thread is created for each kind of fire (persistent or non-persistent).

Connection less client and server:

Below are the readings 10 tries requesting a 1mb file from a connection-less server.
Total bytes in the file:

Try no.		UDP	
	Total bytes sent	Total byte received	Time taken
1	1071254	279591	0.0539936
2	1071254	282862	0.0557049
3	1071254	276041	0.0542635
4	1071254	283530	0.0541049
5	1071254	483594	0.0963944
6	1071254	280032	0.056101
7	1071254	499650	0.0937981
8	1071254	280187	0.0554714
9	1071254	285272	0.0549178
10	1071254	746521	0.140588

We can see, by contrasting with the connection-oriented server, UDP (connection-less) is faster than TCP (connection oriented).

It has been observed that the entire data is not received by the client. Some packets are dropped when the receive buffer is full, as seen above.
Also, in connection less communication, all the packets come in random order whereas TCP ensures that all the data sent by server is received by client and the data order is preserved.

One important observation: In UDP, the server must indicate to client that it has sent all the data and it can stop waiting for the data. If the packet containing the indication from the server is dropped, the client will keep on listening forever and will not end its connection.