Assignment-4 (100 points)

Due on November 23, 2020 23:59

General Instructions:

This is the second of the programming and hands on networking assignment. Make sure to save the work and store the configuration, installations, test scripts, source code etc. that you do in this assignment. This assignment builds on the previous Assignmnet-3. You need to turn in the following:

1. Observation Report (PDF/Document detailing your program output, screenshots indicating your result, and your analysis. Include any instructions or tools you used. Cite all the referenced resources appropriately.
2. Zip folder containing the source code, support scripts that you have implemented and also the instructions to run the program.

You are free to choose the programming language of your choice. I would recommend 'C' for rich programming experience and understanding the low-level system APIs, but even languages like Python, Java, or Go would be fine too.

Persistent or Non-persistent connections. (20 points)

1. Extend/Modify the code you had implemented in Assignment-3 to facilitate the client and server programs that enable to download multiple files on a same/single TCP/UDP connection. Assume the user would want to download all the 5 files (that you had chosen earlier). Single client program would receive the input of 5 files, send the requests for 1 file at a time to the server, but re-use the existing connection made with the server for subsequent files.
    (a) Profile and report the one-time connection setup time, total download time and aggregate throughput; also report the download time and achieved throughput for each file of the files.
    (b) Compare the new results with the earlier achieved results (Individual file download time and throughput number). What inference can you make? Is the persistent connection useful? Is this contrary to what you learnt with Persistent vs Non-Persistent? Try to rea son on your observed results.

    Note: This is not a concurrent server; <Document precisely the changes needed on the client and server end to make the persistent mode work.> Also, try to take around 3-5 measurements to make sure that you have reasonable accuracy for the reported numbers. If you implemented persistent mode in the earlier assignment, then you need to implement a client that operates in non-persistent mode and compare. Or if you had implemented concurrent server, then you need to make the server to be non-concurrent.

Concurrent Servers: Fork vs Threads (TCP) *(30 points)*

2. Implement a server that can handle multiple requests concurrently. You have the following two models to build a concurrent server. ***(30 points)***

   Fork Model: The server program <tcp_server_sc_fork.c> would listen on a port 12345. Upon receiving a request from a client, it will fork a child process and let the child process to handle the data transfer, while the parent process goes back to listen for new connections. ***(5 points)***

   Threads Model: Here, the server <tcp_server_sc_thread.c>instead of forking, spawns a new thread to perform the data transfer, while the initial main thread (parent) will go back to listen for new connections. ***(5 points)***

   Now, evaluate and answer the following (use-case: Download 5 files):

   (c) Using the client (with persistent connection), report the completion time and aggregate throughput for Fork-model vs Thread-model. Compare these results with earlier question 1b. Again, which model works better for you? Try to reason your evaluation results. Also, compare the new results with the earlier achieved results (individual file download time and throughput number). What inference can you make? Does the concurrent server benefit or adds to lot of overhead and latency? Try to reason on your observed results. ***(10 points)***

   (d) Use the concurrent server, and use 5 clients (non-persistent type, implemented in Assignment-3) that request for 1 file each. There should be 5 different concurrent client programs trying to download 1 file each. <Note: You need to script a way to launch these 5 clients simultaneously>. Report the completion time and aggregate throughput for Fork-model vs Thread-model. What do you observe? Which model works better for you? Try to reason your evaluation results. ***(10 points)***

Note: For those, who have not heard of Fork vs Threading, you can refer to the online resources or have additional discussion with Instructor or TAs. You are welcome to use the online material/code and repurpose it for your assignment but do cite in such cases. Try to take around 3-5 measurements to make sure that you have reasonable accuracy for the reported numbers.

Migrating to Mininet. (TCP) *(50 points)*

3. Implement a Mininet network, where you instantiate different desired (parameterized) topologies, where one of the host node acts as server and subset of the remaining nodes act as a clients. First, detail precisely the changes you need to make for the client and server to operate with Mininet (***5 points)***

   Refer ***http://mininet.org/walkthrough/***

   (e) **Single topology (i)**: Make use of the single topology of the mininet that allows you to instantiate 1 switch and 6 host nodes. Designate one of the host nodes to run as server and remaining 5 as clients. All the Clients would reach the server (concurrent mode – prefer to choose the threading model) and download 1 file each. <Now report the download time and throughput for each of these clients. What do you observe? (a) How far are these results when compared to non-mininet version you had implemented earlier in Q2d. (***10 points)***

   $ sudo mn --topo single,6

   (f) **Single topology (ii)**: Here, you reuse the above topology, but modify the argument to the client program so that all the clients try to download the same file. < Now report the download time and throughput for each of these clients. What do you observe? (a) what do you think would be the reason for such variances, if you found any? (***10 points)***

   (g) **Linear Topology (i) (Bandwidth)**: Choose the largest of the file you have used. Now, try to start with a very small linear topology (2 host nodes and 1 switch). In each iteration vary the link bandwidth from 10Mb, 100Mb, and 1Gbps respectively. In each iteration, record the download time and the achieved throughput. Report your observations. And for subsequent questions choose the link that provides the best throughput. (***5 points)***

   $ sudo mn --link tc, bw=10Mb --topo linear,1

   (h) **Linear Topology (ii) (Delay)**: Choose the largest of the file you have used. Now, try to start with a very small linear topology (2 host nodes and 1 switch). In each iteration vary the link delay as 1ms, 2ms, 5ms and 10ms respectively. In each iteration, record the download time and the achieved throughput. Report your observations. (***5 points)***

   $ sudo mn --link tc, delay=1ms, bw=??--topo linear,1

   (i) **Linear Topology (iii) (Loss)**: Choose the largest of the file you have used. Now, try to start with a very small linear topology (2 host nodes and 1 switch). In each iteration vary the link loss percentage as 1%, 2% and 5% respectively. In each iteration, record the download time and the achieved throughput. Report

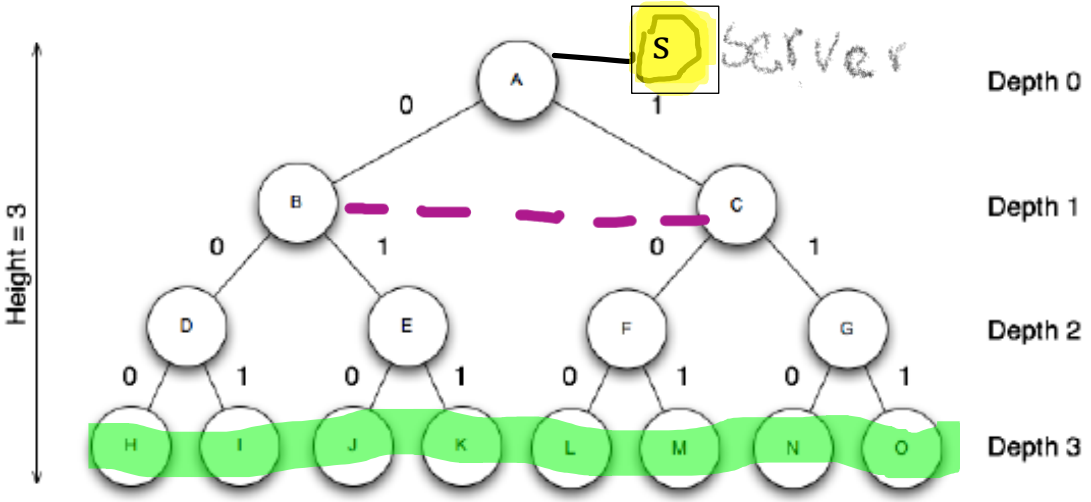
your observations. (***5 points)***

$ sudo mn --link tc, loss=1, bw=?? --topo linear,1

(j) **Linear Topology (iv) (Hops)**: Choose the largest of the file you have used. Now, try to start with a very small linear topology (2 host nodes and 2 switch) and in every iteration you scale the number of switches by 1. Starting from 2 up to 10 switches in steps of 2 (i.e. 2,4,6,8,10). In each case, record the download time and the achieved throughput. Report your observations. (***10 points)***

Grace Question on Mininet (Custom Topology). (TCP/UDP) *(50 points)*

4. Implement a Mininet network, where you instantiate different desired (custom) topologies.
   Refer to https://github.com/mininet/mininet/wiki/Introduction-to-Mininet



(k) **Custom Topology (i) (Varying link Bandwidth)**: Implement a custom fat-tree tree topology with Depth(D)=3 as shown above (ignore the 0/1 markings). All the leaf nodes (marked Green) and node at root (S marked yellow) will be the host elements and rest of the intermediate nodes will be the switch elements. Links between D0 and D1 (AB and AC) will have twice the bandwidth than the Links between D1 and D2 (BD, DB, CF, CG) and likewise at higher depths. Say the Link HD=10Mbps, then BD=20Mbps and AB=40Mbps. Note: Link AS will have same bandwidth as link AB. (***20 points)***

   a. Choose any of earlier 4 files to download, Setup the server at S and clients at nodes H, I, J, and K respectively. What is the total time and aggregate throughput? (***10 points)***
   b. Now, switch the client nodes to be H, K, M and N

respectively. Measure the total time and aggregate throughput. Any differences? Reason. (***10 points)***

(l) **Custom Topology (ii) Horizontal scaling and Load balancing**: Here, scale the number of server instances such that the server program is run at different host nodes (new nodes) and clients choose to communicate with one server each. (***20 points)***

a. Choose the same earlier 4 files to download, Setup the server at S, add two more server (host nodes, say S' and S'' at switches F and G respectively, and the clients at nodes H, I ,J, and K respectively. Map the client servers as H→ S, I→S, ,J→ S', K→S''. What is the total time and aggregate throughput? (***10 points)***

b. Now, switch the client nodes to be H, K, M and N respectively. Map the client servers as H→ S, K→S, ,M→ S', N→S''. Measure the total time and aggregate throughput. Any differences? Reason. (***10 points)***

(m) **Custom Topology (iii) (with Loop)**: Here, add a new link BC (shown with dotted lines in the figure). Now try to repeat the above experiment (4L(a) and 4L(b)). What do you observe? Report your results and analysis. (***10 points)***

This will be your final assignment ☺.