Programming Assignment 2
CS550 - Advanced OS
Fall 2020
Amit Nikam
A20470263
anikam@hawk.iit.edu

EVALUATION REPORT

1) One Node to Node File Transfer:

Establishing a successful distributed hash table server and node connection by defining communication protocol was the first challenge in implementing multithreaded this naive P2P architecture.

For this test, Node A was able to connect to the DHT server andget the DHT Record then connect to Node b download a .txt format file. Total 140 Bytes were transferred in just 0.00032 seconds.

2) 8 Node Interconnected:

The next step was to implement a DHT server and Nodes which can handle multiple clients at the same time. To achieve this, threading was implemented on each of them. Each connection to them is handled through a handler which is threaded. This way multiple connections are handled at the same time.

It was observed that the transfer times were way too less than those in Programming Assignment 1 for this same test which is a good thing. This could be because the server itself was not at load, rather the load was spread across the nodes.

3) Multiple Node - Multiple files:

For this evaluation, 2 files of 140 Bytes each were used. The same process of downloading / uploading these files simultaneously was repeated for incrementing number of client connections.

I established download and upload nodes randomly, so that some nodes might have a chance to fault out, then I calculated the denials to understand the performance. Some faulted nodes had more than one requests and so the denies might be higher than faulted nodes.

In my implementation I have programmed a triple threaded node: (1) Node Handler to keep in sync with DHT Server (2) Client handler to keep listening for incoming connections and message (3) Download Handler to download files.

So a Faulted Node can be calculated as (expected threads - peak threads during implementation)/3. Note that this number could be fractional, indicating some of the nodes partially faulted but recovered.

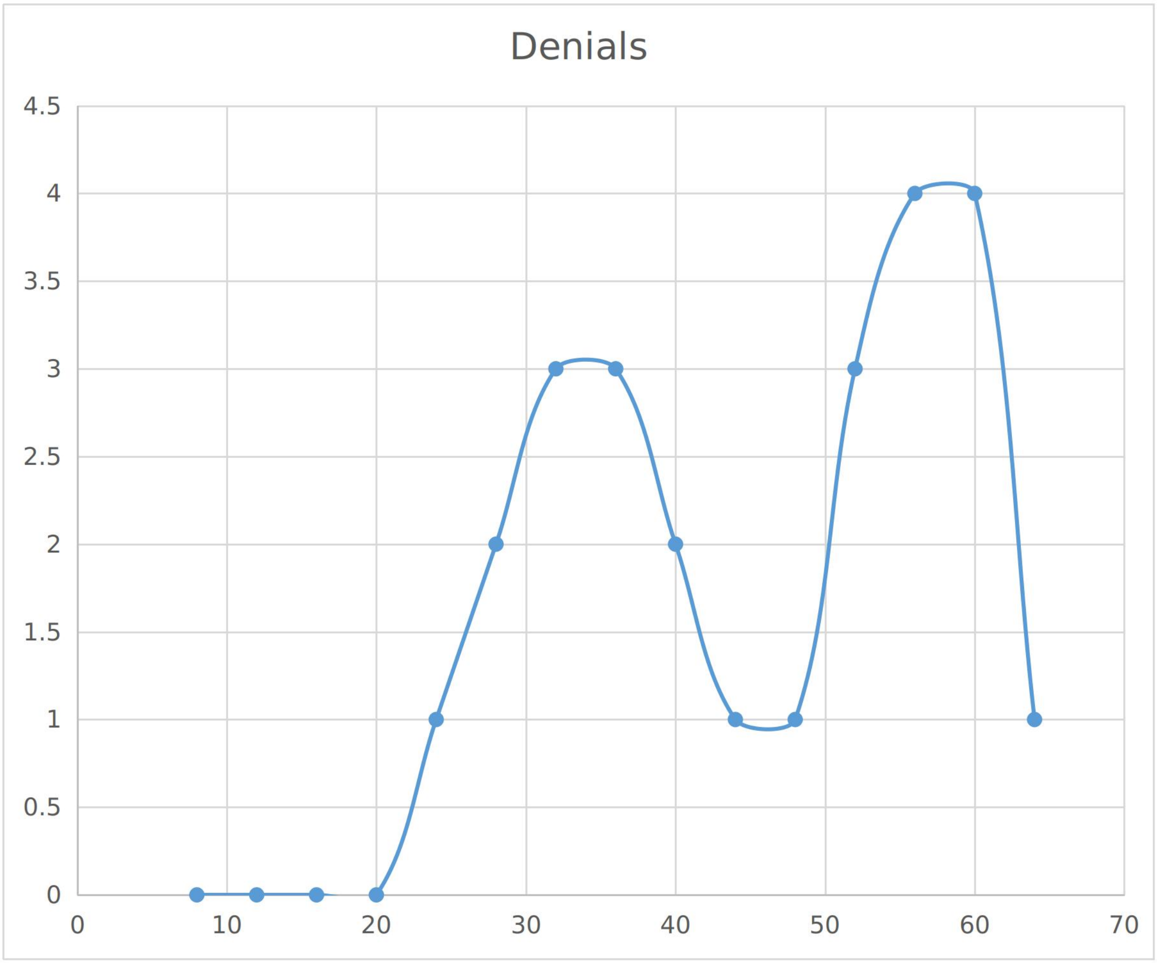
The following table is based on the implementation outcomes

# Client	Total Denies	Peak Threads (PT)	Expected Threads (ET)	Faulted Nodes =(ET-PT)/3
8	0	40	40	0
12	0	60	60	0
16	0	80	80	0
20	0	100	100	0
24	1	117	120	1
28	2	134	140	2
32	3	151	160	3
36	3	167	180	4.34
40	2	194	200	2
44	1	220	220	0
48	1	238	240	0.68
52	3	260	260	0
56	4	277	280	1
60	4	297	300	1
64	1	317	320	1

It can be observed from the above table that nodes were denied service at max 4 times, finding max fault percentage from the above data (3*2/32) = 0.1875 i.e. 18.75% change of denial at 32 connection. 18.75% happens to be the worst scenario among the readings calculated.

Comparing this to Server Client implementation from Programming Assignment 1 which had an average denial rate of 33%, using P2P architecture seems much better.

We would be plotting this outcome with respect to the number of nodes



Conclusion: In the end, a multithreaded client-server nodes and DHT Sever implementation was achieved. Even under heavy loads the system performance was consistent with just an average denial rate of 1.6 denial out of total requests. It is observed that the nodes performed really well at 20 or less nodes in the system with a 0% denial.