Advanced Operating Systems Programming Assignment 2 Final Report

A20491471-Sashank Lakshmana Bommadevara

A20516702 - Abdurakhmon Urazboev

A20519664-Syed Alle Mustafa

Objective:

This assignment's main goal was to create a scaled-down, decentralized, peer-to-peer file-sharing system that is similar to the Gnutella network. The main features we concentrated on were file registration, file querying, and the usage of a flood mechanism for inter-server (or peer) communication.

System Architecture:

Each node, or server, in the system is essentially decentralized, acting both as a client (to send queries to nearby servers) and as a server (to reply to incoming requests).

1. Server Role:

    Peer-to-peer communication, resource discovery, and file sharing are made possible in a decentralized network by this server code. It upholds Gnutella protocol standards, directs search requests, handles connections, and keeps resource indexing. This code lessens the dependency on centralized servers while supporting a fault-tolerant and scalable network.

2. Client Role:

    A server connects to its neighbors' servers and asks for file searches in order to perform the function of a client when it needs to query its neighbors.

Key Functionalities:

1. File Registration:

    File registration is possible for clients with the server they are connected to. The IP address of the client and the files they have registered are stored on the server.When other clients or servers request a search for a specific file, this registration procedure is helpful.

2. File Searching:

    File searches are requests that clients can submit.Prior to attempting to access the requested file, the server searches locally on any of its connected clients. The search request is forwarded by the server utilizing the flood technique to its nearby servers if it cannot be discovered locally and the Time-To-Live (TTL) for the request has not expired.

3. Flood Mechanism:

    Flooding in our architecture refers to pinging all known neighbors with a search request. An individual identification (message_ID) is attached to every message or query that travels over the network. With the use of this identifier, the same query cannot be propagated in a circular fashion. Circular loops and needless network congestion are prevented when a server recognizes a message ID and refuses to process it again.

Design Considerations & Optimizations:

1. Threaded Implementation:

We used a multi-threaded strategy in order to manage several clients and servers at the same time. In order to prevent processes from stopping and to maximize system throughput, each new connection generates a new thread.

2. Data Structures:

Which client IP has registered which files are tracked using a concurrent hash map. Considering our concurrent model, thread-safe operations are provided by this. To prevent duplicate processing, the server uses an array list to keep track of the message IDs it has handled.

3. Configuration-Driven Approach:

The system is scalable and versatile because a configuration file (configFileName) makes it simple to design and modify the behavior of servers, particularly their surrounding relationships.

Challenges & Limitations:

1. Scalability:

As the network expands, the flooding mechanism may become bandwidth-intensive even if it guarantees that queries reach every section of the network. For really big networks, this method might not function well.

2. TTL Management:

   Determining the right TTL for search requests is essential. A request may not reach possible servers having the necessary file if the TTL is too short. A very high TTL, on the other hand, may result in an excessive amount of network traffic.

3. Duplicate File Entries:

   Cases where more than one client may have the same file are not supported by the current system design. To guarantee quicker download times for consumers, an addon to our system might assign a server ranking according to criteria such as connection speed.

Results:

For centralized topology: 1 indexing server, 16 peers and 16 clients connected to them

Experiment #1:

1 peer issuing 10K requests:

Peer#	Search time
1	55 milliseconds

Experiment #2:

9 peers connected issuing 90k requests:

Peer#	Search Time
1	335.67 secs
2	338.32 secs
3	333.12 secs
4	326.45 secs
5	337.10 secs
6	336.30 secs
7	329.26 secs
8	332.63 secs
9	342. 48 secs
Avg	334.58 secs

Experiment #3:

9 peers issuing 90k requests and obtaining 10KB the file:

Peer#	Search and obtain time
1	562.36 seconds
2	585.12 seconds
3	552.49.seconds
4	554.89 seconds
5	609.11 seconds
6	521.93 seconds
7	572.02 seconds
8	629.67 seconds
9	553.89 seconds
Average	571.27 seconds

Experiment #4:

9 peers issuing 10 requests and obtaining 100MB files:

Peer#	Search and obtain time
1	15.62 seconds
2	14.26 seconds
3	16.84.seconds
4	15.23 seconds
5	18.97 seconds
6	12.55 seconds
7	16.34 seconds
8	9.67 seconds
9	12.13 seconds
Average	14.62 seconds

Results for Star Topology:

Experiment #1:

1 peer issuing 10K requests:

Peer#	Search time
1	50 milliseconds

Experiment #2:

9 peers connected issuing 90k requests:

Peer#	Search Time
1	309.16 secs
2	331.65 secs
3	312.91 secs
4	293.66 secs
5	320.11 secs
6	302.43 secs
7	339.72 secs
8	298.58 secs
9	311.89 secs
Avg	313.34secs

Experiment #3:

9 peers issuing 90k requests and obtaining 10KB the file:

Peer#	Search and obtain time
1	540.03 seconds
2	543.51 seconds

3	541.36 seconds
4	548.47 seconds
5	551.13 seconds
6	541.87 seconds
7	549.21 seconds
8	546.43 seconds
9	550.09 seconds
Average	551.34 seconds

Experiment #4:

9 peers issuing 10 requests and obtaining 100MB files:

Peer#	Search and obtain time
1	13.65 seconds
2	14.89 seconds
3	15.23.seconds
4	17.2 seconds
5	11.65 seconds
6	16.28 seconds
7	14.67 seconds
8	10.10 seconds
9	11.63 seconds
Average	13.92 seconds

Results for 2D Mesh Topology:

Experiment #1:

1 peer issuing 10K requests:

Peer#	Search time
1	41 milliseconds

Experiment #2:

9 peers connected issuing 90k requests:

Peer#	Search Time
1	290.16 secs
2	292.38 secs
3	296.52 secs
4	298.17 secs
5	293.68 secs
6	291.92 secs
7	302.45 secs
8	292.57 secs
9	286.19 secs
Avg	296.94

Experiment #3:

9 peers issuing 90k requests and obtaining 10KB the file:

Peer#	Search and obtain time
1	515.49 seconds
2	521.21 seconds
3	587.12.seconds
4	503.93 seconds
5	496.37 seconds
6	523.68 seconds
7	519.55 seconds
8	531.07 seconds
9	517.29 seconds
Average	523.96 seconds

Experiment #4:

9 peers issuing 10 requests and obtaining 100MB files:

Peer#	Search and obtain time
1	9.28 seconds
2	11.64 seconds
3	11.25.seconds
4	14.92 seconds

5	9.65 seconds
6	13.16 seconds
7	12.59 seconds
8	11.82 seconds
9	10.47 seconds
Average	11.63 seconds

Conclusion:

In summary, the Gnutella network we implemented showed that the 2D topology outperformed the centralized and star topologies in terms of efficiency and speed. The 2D topology's decentralized structure facilitated quicker resource identification and file exchange. This study underlines the potential of 2D topologies for peer-to-peer networks and underscores the significance of network topology in performance optimization. These subtleties should be investigated in future research to improve P2P network dependability and efficiency. Although we see very small differences in terms of numbers, we should keep in mind that this is a small experimental setup and differences in real life scenarios with bigger networks can show completely different results. Also the sequence of requests that the receiving server sends out to peers also affects the results depending on the number of misses.

The End