Tanmay Pramanick – A20541164

PA2 – Peer-to-peer publisher subscriber system

Report:

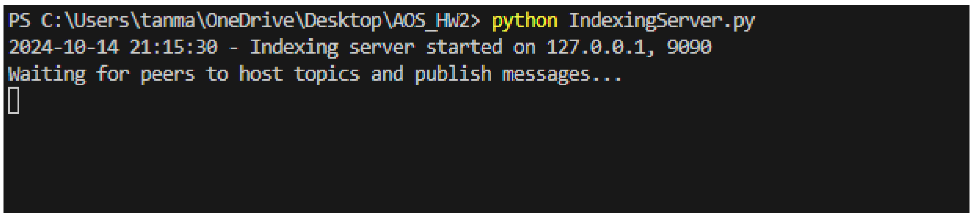
Note:

- To ensure proper execution, each component of the system needs to be run in a separate terminal. This setup allows for the independent operation of the Indexing Server, Peer Nodes, Publishers, and Subscribers.
- Instead of using automated scripts like Makefile or Ant, we opted for a simpler approach where each Python file is run manually. This allows for flexibility when configuring multiple peers, as peer IPs and ports are specified via command-line arguments. Given the dynamic nature of specifying peers, automating this process would limit our ability to manage and scale the peers as needed, making manual execution more straightforward and effective for this assignment.
- This program is running successfully in Linux Environment.

Starting the Program:

Step 1: Running the Indexing Server

To begin, we start by running **IndexingServer.py**. This server is the central point for maintaining a registry of available topics and peer nodes.

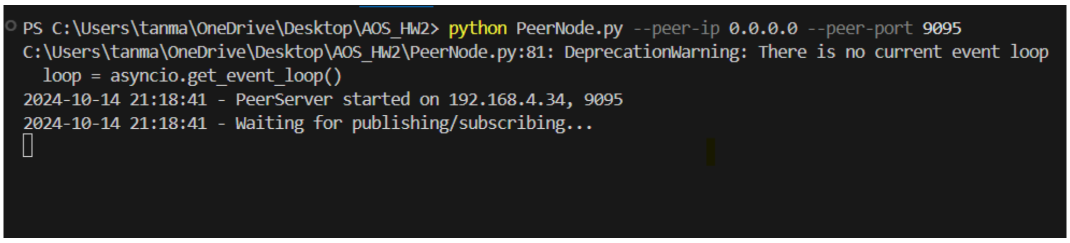


```
PS C:\Users\tanma\OneDrive\Desktop\AOS_HW2> python IndexingServer.py
2024-10-14 21:15:30 - Indexing server started on 127.0.0.1, 9090
Waiting for peers to host topics and publish messages...
```

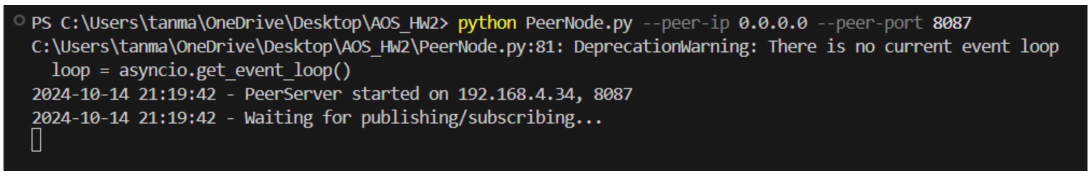
Step 2: Running Peer Nodes

Next, we start the peer nodes using **PeerNode.py**, specifying their **IP** and **port**.

In this example, we run two instances of **PeerNode.py** to demonstrate the capability of handling multiple peers and publishers. This setup illustrates the robustness and scalability of the code.



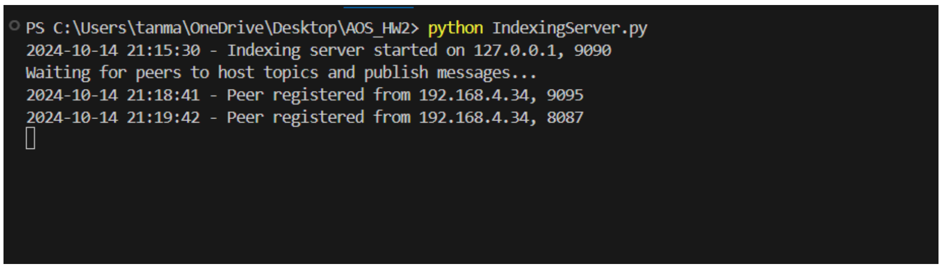
```
PS C:\Users\tanma\OneDrive\Desktop\AOS_HW2> python PeerNode.py --peer-ip 0.0.0.0 --peer-port 9095
C:\Users\tanma\OneDrive\Desktop\AOS_HW2\PeerNode.py:81: DeprecationWarning: There is no current event loop
  loop = asyncio.get_event_loop()
2024-10-14 21:18:41 - PeerServer started on 192.168.4.34, 9095
2024-10-14 21:18:41 - Waiting for publishing/subscribing...
```



```
PS C:\Users\tanma\OneDrive\Desktop\AOS_HW2> python PeerNode.py --peer-ip 0.0.0.0 --peer-port 8087
C:\Users\tanma\OneDrive\Desktop\AOS_HW2\PeerNode.py:81: DeprecationWarning: There is no current event loop
  loop = asyncio.get_event_loop()
2024-10-14 21:19:42 - PeerServer started on 192.168.4.34, 8087
2024-10-14 21:19:42 - Waiting for publishing/subscribing...
```

In the above snippet, I have used IP as 0.0.0.0 as the IP allows the peer to bind to any available IP address, enabling flexibility across different environments.

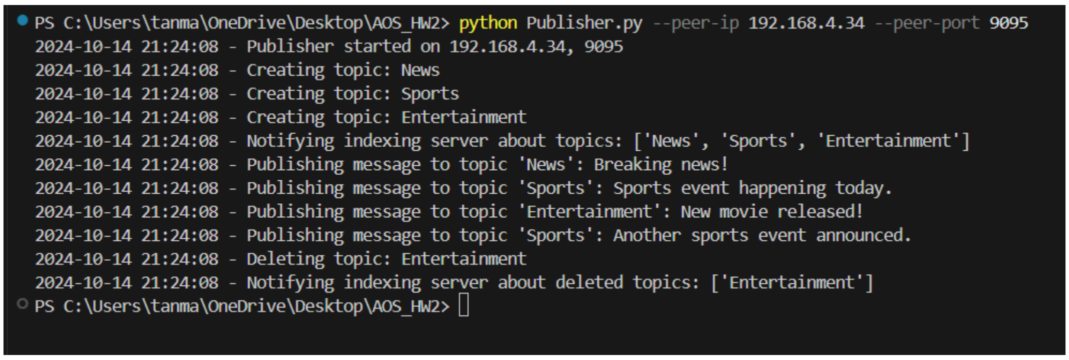
Now, you can see two peers are registered in the Indexing Server Output:



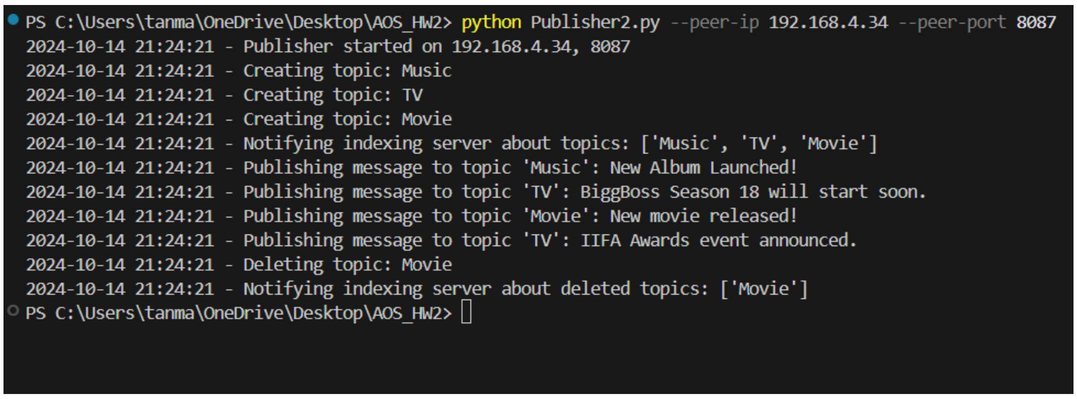
```
PS C:\Users\tanma\OneDrive\Desktop\AOS_HW2> python IndexingServer.py
2024-10-14 21:15:30 - Indexing server started on 127.0.0.1, 9090
Waiting for peers to host topics and publish messages...
2024-10-14 21:18:41 - Peer registered from 192.168.4.34, 9095
2024-10-14 21:19:42 - Peer registered from 192.168.4.34, 8087
```

Step 3: Running Publisher Clients

After setting up the peers, we launch our publisher clients (**Publisher.py** and **Publisher1.py**). These publishers connect to the registered peers by specifying the corresponding IP and port of the PeerNode.

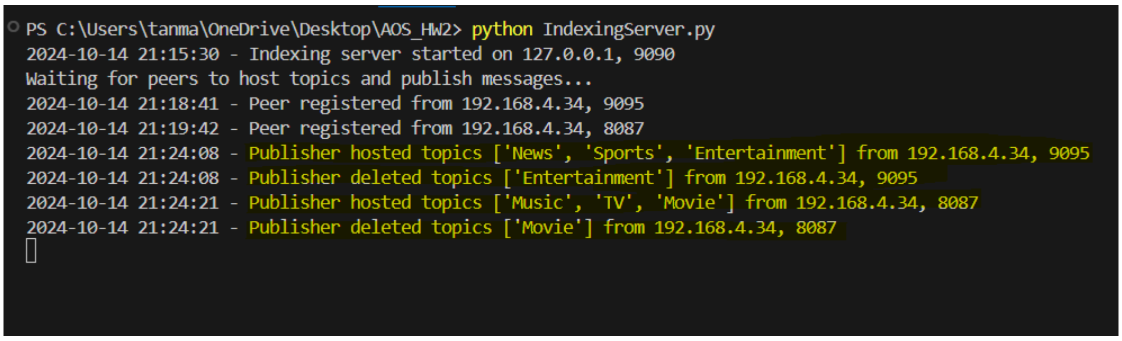


```
PS C:\Users\tanma\OneDrive\Desktop\AOS_HW2> python Publisher.py --peer-ip 192.168.4.34 --peer-port 9095
2024-10-14 21:24:08 - Publisher started on 192.168.4.34, 9095
2024-10-14 21:24:08 - Creating topic: News
2024-10-14 21:24:08 - Creating topic: Sports
2024-10-14 21:24:08 - Creating topic: Entertainment
2024-10-14 21:24:08 - Notifying indexing server about topics: ['News', 'Sports', 'Entertainment']
2024-10-14 21:24:08 - Publishing message to topic 'News': Breaking news!
2024-10-14 21:24:08 - Publishing message to topic 'Sports': Sports event happening today.
2024-10-14 21:24:08 - Publishing message to topic 'Entertainment': New movie released!
2024-10-14 21:24:08 - Publishing message to topic 'Sports': Another sports event announced.
2024-10-14 21:24:08 - Deleting topic: Entertainment
2024-10-14 21:24:08 - Notifying indexing server about deleted topics: ['Entertainment']
PS C:\Users\tanma\OneDrive\Desktop\AOS_HW2>
```



```
PS C:\Users\tanma\OneDrive\Desktop\AOS_HW2> python Publisher2.py --peer-ip 192.168.4.34 --peer-port 8087
2024-10-14 21:24:21 - Publisher started on 192.168.4.34, 8087
2024-10-14 21:24:21 - Creating topic: Music
2024-10-14 21:24:21 - Creating topic: TV
2024-10-14 21:24:21 - Creating topic: Movie
2024-10-14 21:24:21 - Notifying indexing server about topics: ['Music', 'TV', 'Movie']
2024-10-14 21:24:21 - Publishing message to topic 'Music': New Album Launched!
2024-10-14 21:24:21 - Publishing message to topic 'TV': BiggBoss Season 18 will start soon.
2024-10-14 21:24:21 - Publishing message to topic 'Movie': New movie released!
2024-10-14 21:24:21 - Publishing message to topic 'TV': IIFA Awards event announced.
2024-10-14 21:24:21 - Deleting topic: Movie
2024-10-14 21:24:21 - Notifying indexing server about deleted topics: ['Movie']
PS C:\Users\tanma\OneDrive\Desktop\AOS_HW2>
```

As seen above, the publishers have successfully hosted and deleted topics on the peers, which can be verified in the Indexing Server's terminal output.



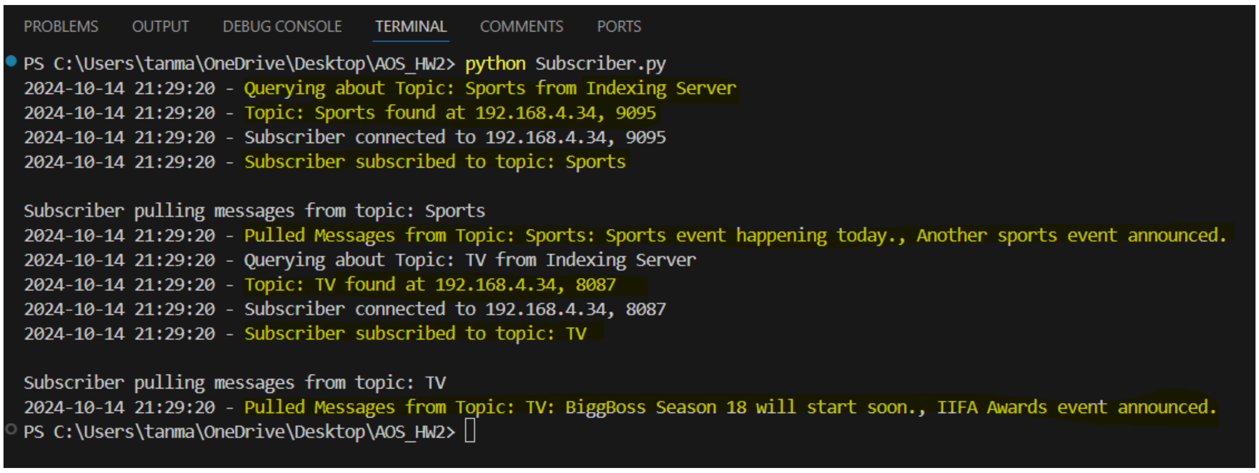
```
PS C:\Users\tanma\OneDrive\Desktop\AOS_HW2> python IndexingServer.py
2024-10-14 21:15:30 - Indexing server started on 127.0.0.1, 9090
Waiting for peers to host topics and publish messages...
2024-10-14 21:18:41 - Peer registered from 192.168.4.34, 9095
2024-10-14 21:19:42 - Peer registered from 192.168.4.34, 8087
2024-10-14 21:24:08 - Publisher hosted topics ['News', 'Sports', 'Entertainment'] from 192.168.4.34, 9095
2024-10-14 21:24:08 - Publisher deleted topics ['Entertainment'] from 192.168.4.34, 9095
2024-10-14 21:24:21 - Publisher hosted topics ['Music', 'TV', 'Movie'] from 192.168.4.34, 8087
2024-10-14 21:24:21 - Publisher deleted topics ['Movie'] from 192.168.4.34, 8087
```

Step 4: Running the Subscriber Client

Now, we run the **Subscriber.py** client to subscribe to topics and pull messages.

The subscriber starts by querying the Indexing Server for available topics.

The Indexing Server responds with the IP and port of the peer hosting that topic, allowing the subscriber to connect and subscribe or pull messages from that topic.



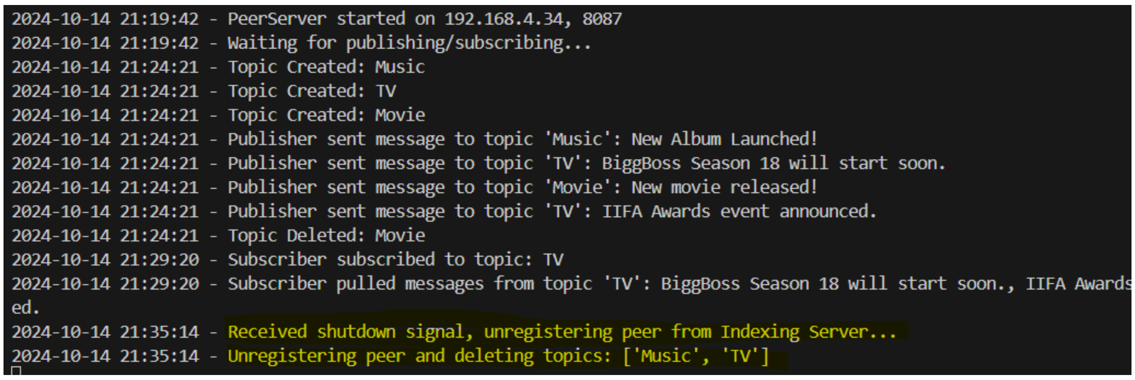
```
PS C:\Users\tanma\OneDrive\Desktop\AOS_HW2> python Subscriber.py
2024-10-14 21:29:20 - Querying about Topic: Sports from Indexing Server
2024-10-14 21:29:20 - Topic: Sports found at 192.168.4.34, 9095
2024-10-14 21:29:20 - Subscriber connected to 192.168.4.34, 9095
2024-10-14 21:29:20 - Subscriber subscribed to topic: Sports

Subscriber pulling messages from topic: Sports
2024-10-14 21:29:20 - Pulled Messages from Topic: Sports: Sports event happening today., Another sports event announced.
2024-10-14 21:29:20 - Querying about Topic: TV from Indexing Server
2024-10-14 21:29:20 - Topic: TV found at 192.168.4.34, 8087
2024-10-14 21:29:20 - Subscriber connected to 192.168.4.34, 8087
2024-10-14 21:29:20 - Subscriber subscribed to topic: TV

Subscriber pulling messages from topic: TV
2024-10-14 21:29:20 - Pulled Messages from Topic: TV: BiggBoss Season 18 will start soon., IIFA Awards event announced.
PS C:\Users\tanma\OneDrive\Desktop\AOS_HW2>
```

This output shows the subscriber successfully connecting to the specified peer, subscribing to the topic, and pulling messages.

Step 5: Shutting Down Peer Nodes

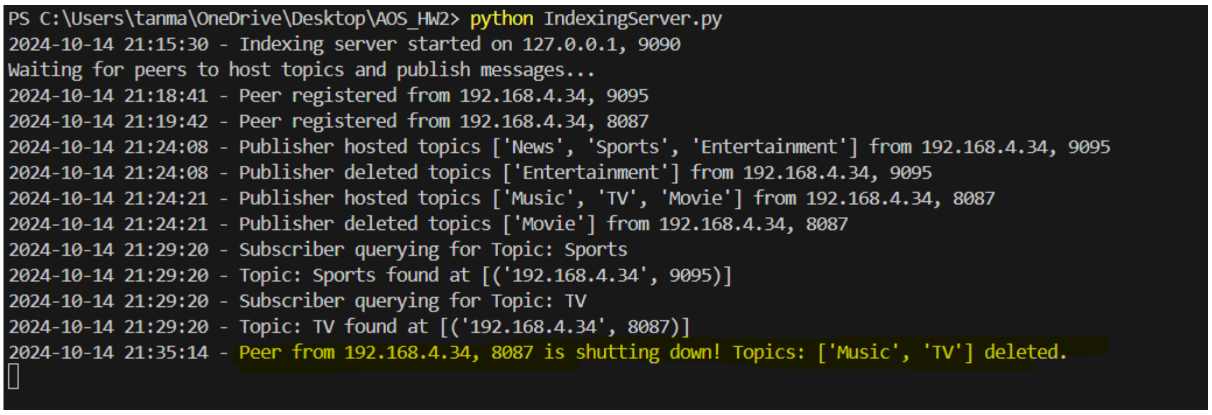
If a PeerNode needs to shut down, it can be done by pressing **Ctrl + C** in its terminal.

When this occurs, the PeerNode sends a shutdown message to the Indexing Server, which then unregisters the peer and deletes the topics hosted by that peer.



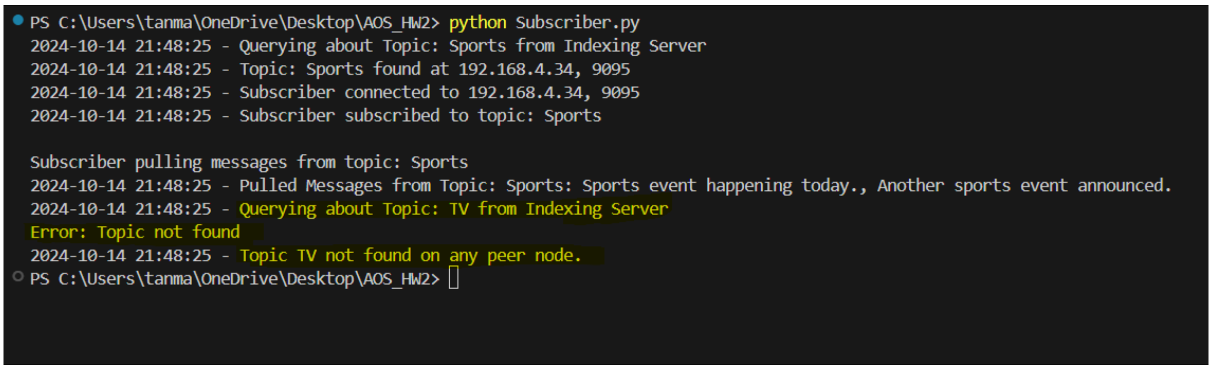
```
2024-10-14 21:19:42 - PeerServer started on 192.168.4.34, 8087
2024-10-14 21:19:42 - Waiting for publishing/subscribing...
2024-10-14 21:24:21 - Topic Created: Music
2024-10-14 21:24:21 - Topic Created: TV
2024-10-14 21:24:21 - Topic Created: Movie
2024-10-14 21:24:21 - Publisher sent message to topic 'Music': New Album Launched!
2024-10-14 21:24:21 - Publisher sent message to topic 'TV': BiggBoss Season 18 will start soon.
2024-10-14 21:24:21 - Publisher sent message to topic 'Movie': New movie released!
2024-10-14 21:24:21 - Publisher sent message to topic 'TV': IIFA Awards event announced.
2024-10-14 21:24:21 - Topic Deleted: Movie
2024-10-14 21:29:20 - Subscriber subscribed to topic: TV
2024-10-14 21:29:20 - Subscriber pulled messages from topic 'TV': BiggBoss Season 18 will start soon., IIFA Awards
ed.
2024-10-14 21:35:14 - Received shutdown signal, unregistering peer from Indexing Server...
2024-10-14 21:35:14 - Unregistering peer and deleting topics: ['Music', 'TV']
```

And in the Indexing Server output:



```
PS C:\Users\tanma\OneDrive\Desktop\AOS_HW2> python IndexingServer.py
2024-10-14 21:15:30 - Indexing server started on 127.0.0.1, 9090
Waiting for peers to host topics and publish messages...
2024-10-14 21:18:41 - Peer registered from 192.168.4.34, 9095
2024-10-14 21:19:42 - Peer registered from 192.168.4.34, 8087
2024-10-14 21:24:08 - Publisher hosted topics ['News', 'Sports', 'Entertainment'] from 192.168.4.34, 9095
2024-10-14 21:24:08 - Publisher deleted topics ['Entertainment'] from 192.168.4.34, 9095
2024-10-14 21:24:21 - Publisher hosted topics ['Music', 'TV', 'Movie'] from 192.168.4.34, 8087
2024-10-14 21:24:21 - Publisher deleted topics ['Movie'] from 192.168.4.34, 8087
2024-10-14 21:29:20 - Subscriber querying for Topic: Sports
2024-10-14 21:29:20 - Topic: Sports found at [('192.168.4.34', 9095)]
2024-10-14 21:29:20 - Subscriber querying for Topic: TV
2024-10-14 21:29:20 - Topic: TV found at [('192.168.4.34', 8087)]
2024-10-14 21:35:14 - Peer from 192.168.4.34, 8087 is shutting down! Topics: ['Music', 'TV'] deleted.
```

After shutting down a peer, attempting to subscribe or pull messages from the deleted topic results in an error, as shown below.



```
PS C:\Users\tanma\OneDrive\Desktop\AOS_HW2> python Subscriber.py
2024-10-14 21:48:25 - Querying about Topic: Sports from Indexing Server
2024-10-14 21:48:25 - Topic: Sports found at 192.168.4.34, 9095
2024-10-14 21:48:25 - Subscriber connected to 192.168.4.34, 9095
2024-10-14 21:48:25 - Subscriber subscribed to topic: Sports

Subscriber pulling messages from topic: Sports
2024-10-14 21:48:25 - Pulled Messages from Topic: Sports: Sports event happening today., Another sports event announced.
2024-10-14 21:48:25 - Querying about Topic: TV from Indexing Server
Error: Topic not found
2024-10-14 21:48:25 - Topic TV not found on any peer node.
PS C:\Users\tanma\OneDrive\Desktop\AOS_HW2>
```

The above steps successfully demonstrate the workflow of the system, from starting nodes and registering topics to handling publishers, subscribers, and gracefully shutting down peers.

Starting Testing and Evaluation

With the setup complete, we now move on to evaluate the system by running our testing suite, ensuring that all APIs work correctly under different conditions and that the system performs efficiently under concurrent loads.

All test files are in the folder named **"tests."**

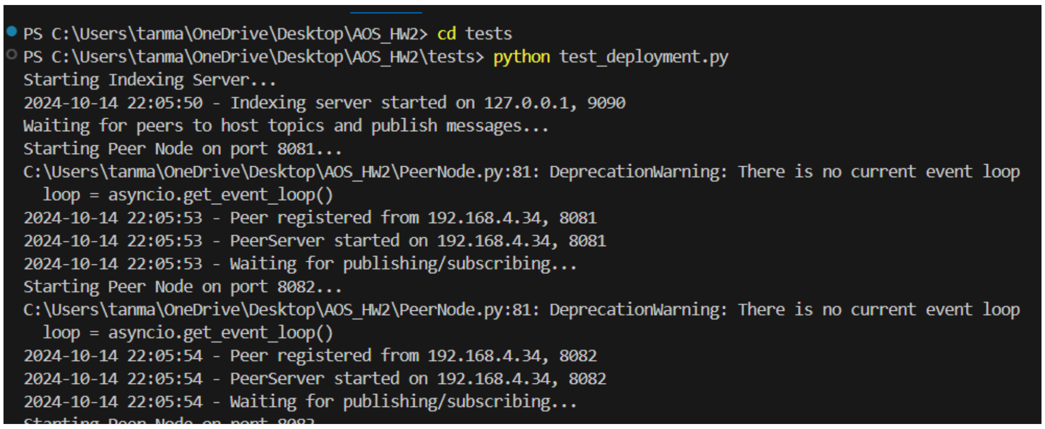
Testing 1: Deployment of Peers and Indexing Server

For this evaluation, we used the script **test_deployment.py**.

We deployed three peer nodes and one Indexing Server on the same machine.

This test verified that all APIs (create topic, delete topic, publish, subscribe, pull) were working correctly.

It also confirmed that multiple peer nodes could simultaneously publish and subscribe to topics without any issues, demonstrating the system's robustness in handling concurrent operations.



```
PS C:\Users\tanma\OneDrive\Desktop\AOS_HW2> cd tests
PS C:\Users\tanma\OneDrive\Desktop\AOS_HW2\tests> python test_deployment.py
Starting Indexing Server...
2024-10-14 22:05:50 - Indexing server started on 127.0.0.1, 9090
Waiting for peers to host topics and publish messages...
Starting Peer Node on port 8081...
C:\Users\tanma\OneDrive\Desktop\AOS_HW2\PeerNode.py:81: DeprecationWarning: There is no current event loop
  loop = asyncio.get_event_loop()
2024-10-14 22:05:53 - Peer registered from 192.168.4.34, 8081
2024-10-14 22:05:53 - PeerServer started on 192.168.4.34, 8081
2024-10-14 22:05:53 - Waiting for publishing/subscribing...
Starting Peer Node on port 8082...
C:\Users\tanma\OneDrive\Desktop\AOS_HW2\PeerNode.py:81: DeprecationWarning: There is no current event loop
  loop = asyncio.get_event_loop()
2024-10-14 22:05:54 - Peer registered from 192.168.4.34, 8082
2024-10-14 22:05:54 - PeerServer started on 192.168.4.34, 8082
2024-10-14 22:05:54 - Waiting for publishing/subscribing...
```

Testing 2: Performance Evaluation

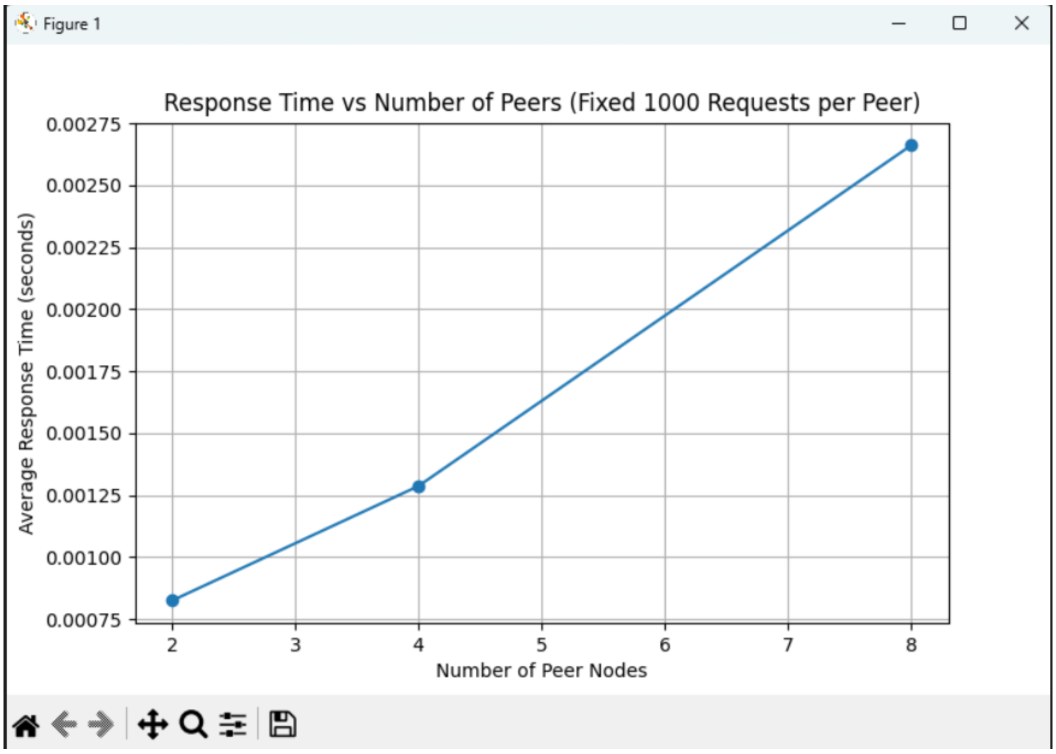
For the performance evaluation, we used the script **test_performance.py** to measure the average response time when multiple peer nodes concurrently queried topics from the Indexing Server.

We varied the number of concurrent peer nodes **(2, 4, 8)** and observed how the average response time changed for each configuration.

Each peer node made 1000 requests.

The results were analysed, and a graph was plotted to demonstrate the system's scalability and response under different loads.

This test validated the efficiency of the Indexing Server in managing increased concurrent peer requests.



Testing 3: Scalability with Large Number of Topics

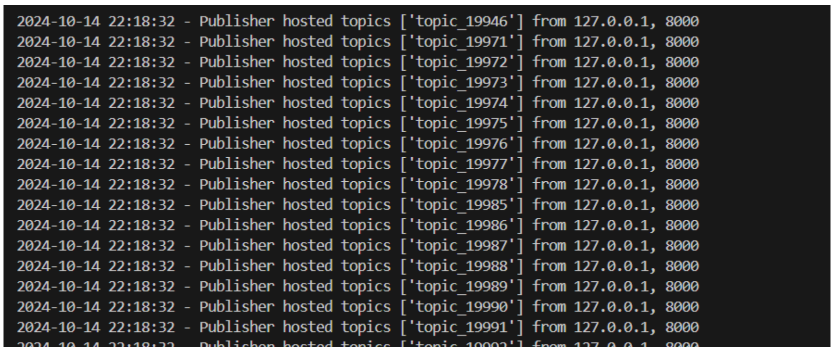
The scalability of the Indexing Server was tested to handle a substantial number of topics.

Instead of using 1 million topics due to the time it required, we opted for **100,000** topics for efficient testing.

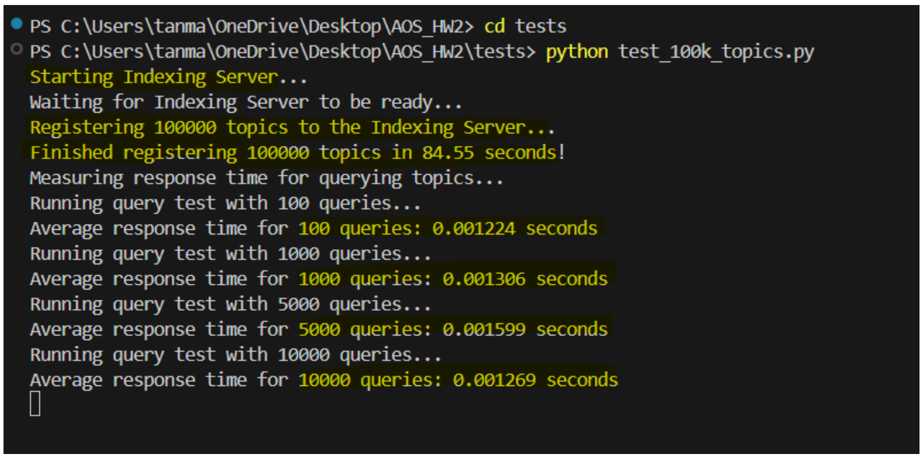
This was implemented in the script **test_100k_topics.py**, located in the tests folder.

The test involved registering a large number of topics and measuring the server's response times for various operations.

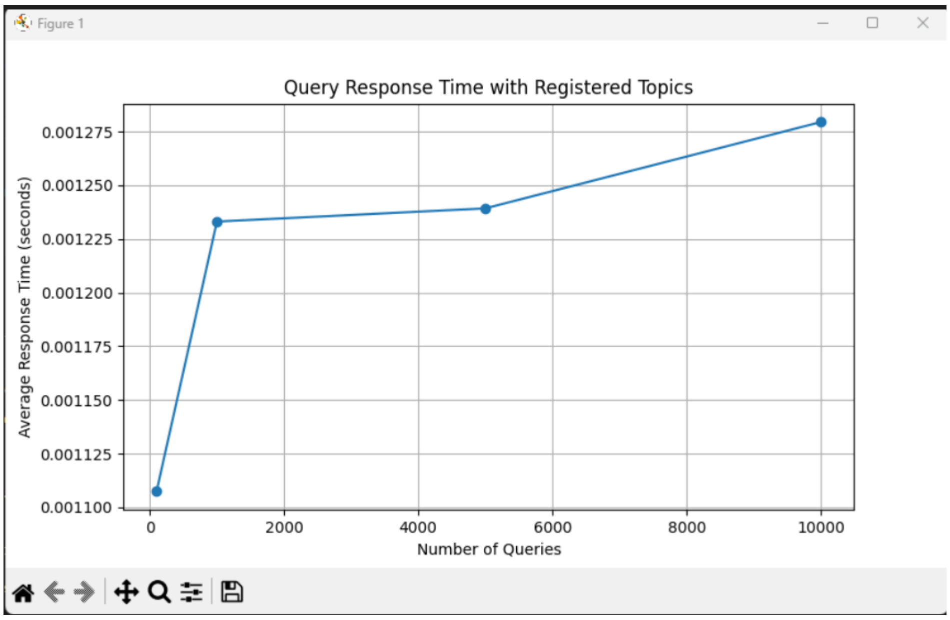
The results were plotted to demonstrate the system's performance in handling many topics, ensuring the design can scale to meet high demands while maintaining acceptable performance.



```
2024-10-14 22:18:32 - Publisher hosted topics ['topic_19946'] from 127.0.0.1, 8000
2024-10-14 22:18:32 - Publisher hosted topics ['topic_19971'] from 127.0.0.1, 8000
2024-10-14 22:18:32 - Publisher hosted topics ['topic_19972'] from 127.0.0.1, 8000
2024-10-14 22:18:32 - Publisher hosted topics ['topic_19973'] from 127.0.0.1, 8000
2024-10-14 22:18:32 - Publisher hosted topics ['topic_19974'] from 127.0.0.1, 8000
2024-10-14 22:18:32 - Publisher hosted topics ['topic_19975'] from 127.0.0.1, 8000
2024-10-14 22:18:32 - Publisher hosted topics ['topic_19976'] from 127.0.0.1, 8000
2024-10-14 22:18:32 - Publisher hosted topics ['topic_19977'] from 127.0.0.1, 8000
2024-10-14 22:18:32 - Publisher hosted topics ['topic_19978'] from 127.0.0.1, 8000
2024-10-14 22:18:32 - Publisher hosted topics ['topic_19985'] from 127.0.0.1, 8000
2024-10-14 22:18:32 - Publisher hosted topics ['topic_19986'] from 127.0.0.1, 8000
2024-10-14 22:18:32 - Publisher hosted topics ['topic_19987'] from 127.0.0.1, 8000
2024-10-14 22:18:32 - Publisher hosted topics ['topic_19988'] from 127.0.0.1, 8000
2024-10-14 22:18:32 - Publisher hosted topics ['topic_19989'] from 127.0.0.1, 8000
2024-10-14 22:18:32 - Publisher hosted topics ['topic_19990'] from 127.0.0.1, 8000
2024-10-14 22:18:32 - Publisher hosted topics ['topic_19991'] from 127.0.0.1, 8000
```



```
PS C:\Users\tanma\OneDrive\Desktop\AOS_HW2> cd tests
PS C:\Users\tanma\OneDrive\Desktop\AOS_HW2\tests> python test_100k_topics.py
Starting Indexing Server...
Waiting for Indexing Server to be ready...
Registering 100000 topics to the Indexing Server...
Finished registering 100000 topics in 84.55 seconds!
Measuring response time for querying topics...
Running query test with 100 queries...
Average response time for 100 queries: 0.001224 seconds
Running query test with 1000 queries...
Average response time for 1000 queries: 0.001306 seconds
Running query test with 5000 queries...
Average response time for 5000 queries: 0.001599 seconds
Running query test with 10000 queries...
Average response time for 10000 queries: 0.001269 seconds
```



Testing 4: Benchmarking Latency and Throughput of Each API

The benchmarking of the APIs for the Peer-to-Peer system was conducted in two scenarios:

1. **With 1 Peer and 1 Indexing Server**: The APIs were benchmarked to calculate the latency and throughput when interacting with a single peer.
2. **With Up to 8 Peers**: The number of peers was gradually increased up to 8, and each API was benchmarked again to measure the impact on latency and throughput.

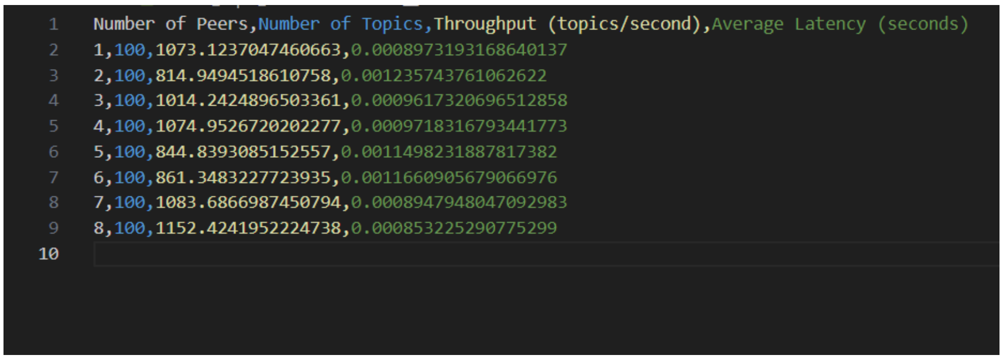
This testing was done using multiple benchmark scripts such as **benchmark_create_topic.py**, **benchmark_delete_topic.py**, **benchmark_send_message.py**, and others.

Each script evaluated the performance of a specific API (create topic, delete topic, send message, subscribe, and pull message). The throughput and latency for each API were measured and plotted into graphs.

The results are available in the graphs folder, with each graph showing the performance as the number of peers increased. The output data is saved in CSV format in the data folder, giving detailed insights into how the system scales and the efficiency of the APIs under varying loads.

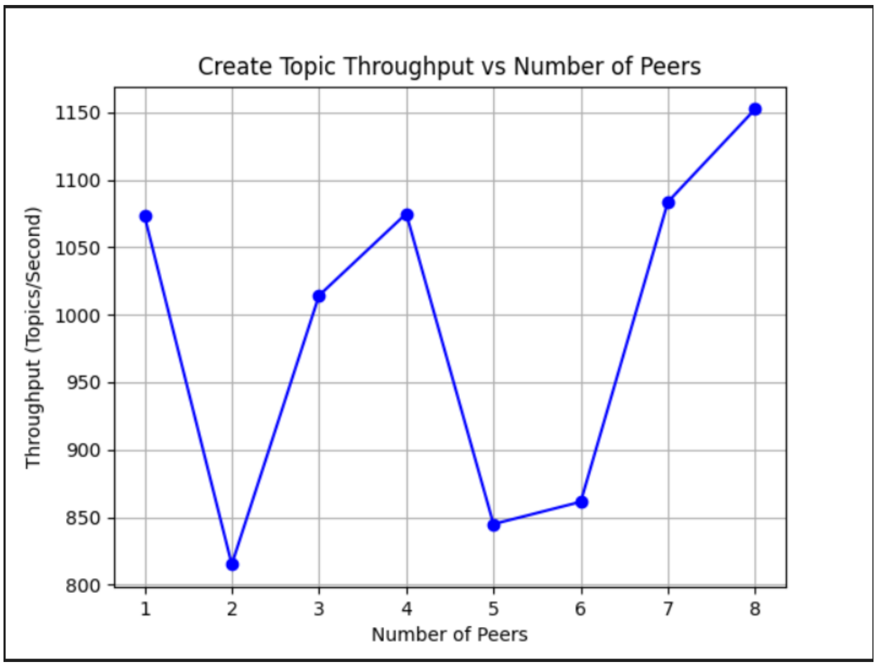
- **Create Topic Benchmark:**

Output in CSV format:

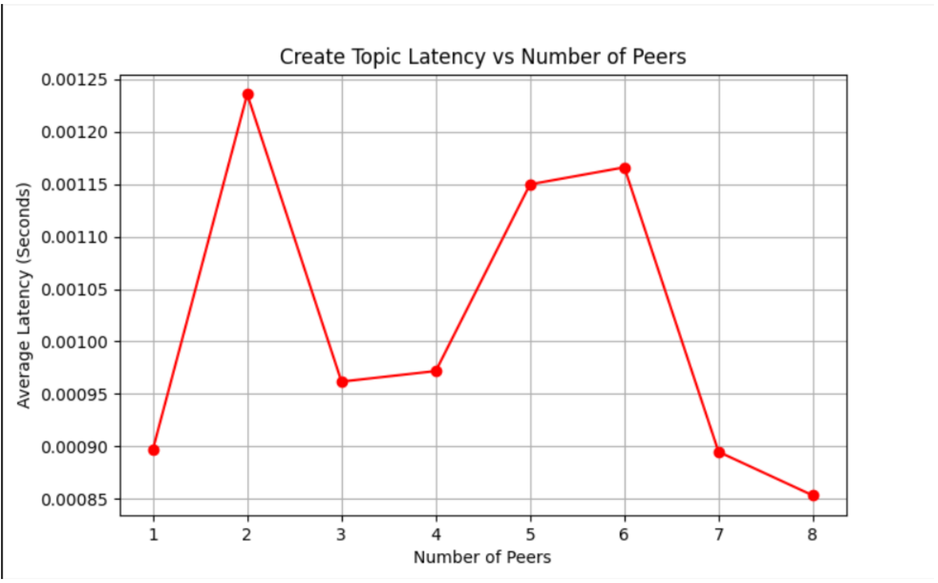


```
Number of Peers,Number of Topics,Throughput (topics/second),Average Latency (seconds)
1,100,1073.1237047460663,0.0008973193168640137
2,100,814.9494518610758,0.001235743761062622
3,100,1014.2424896503361,0.0009617320696512858
4,100,1074.9526720202277,0.0009718316793441773
5,100,844.8393085152557,0.0011498231887817382
6,100,861.3483227723935,0.0011660905679066976
7,100,1083.6866987450794,0.0008947948047092983
8,100,1152.4241952224738,0.000853225290775299
```

Throughput:

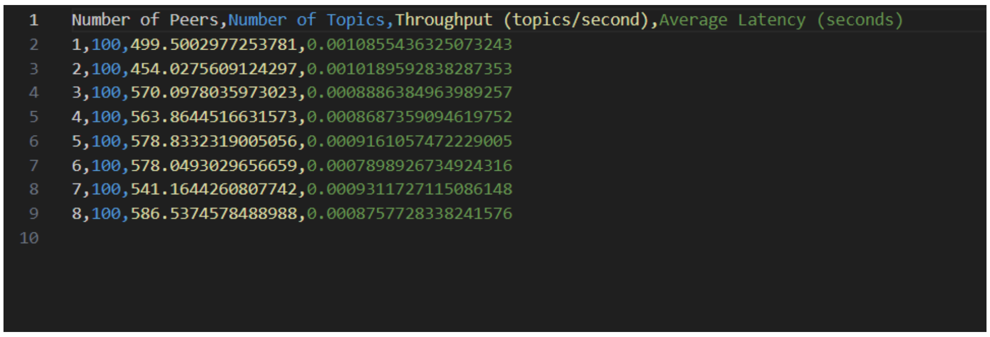


Latency:



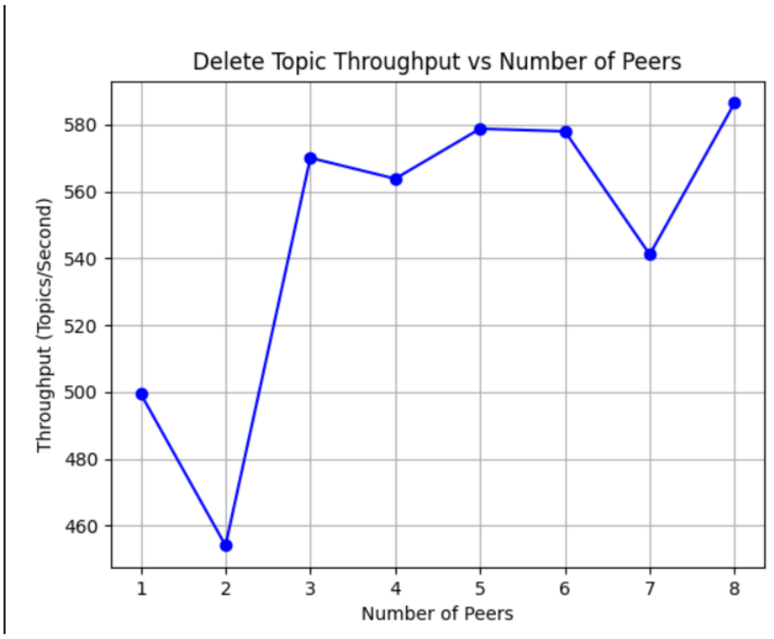
- **Delete Topic Benchmark:**

  Output in CSV format:

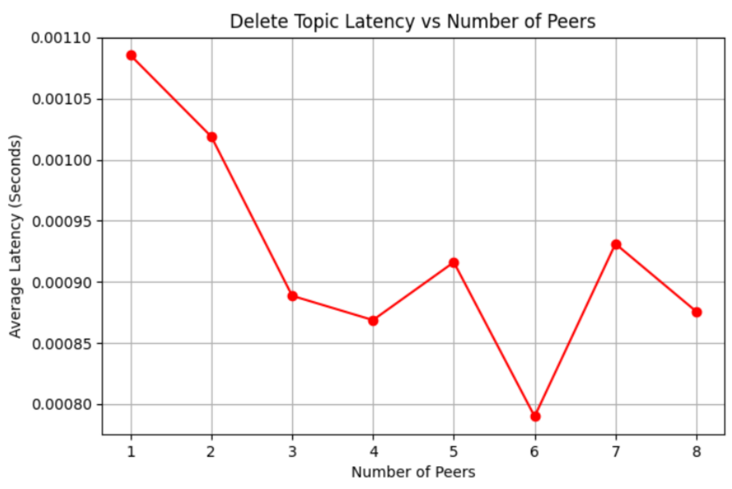


```
Number of Peers,Number of Topics,Throughput (topics/second),Average Latency (seconds)
1,100,499.5002977253781,0.0010855436325073243
2,100,454.0275609124297,0.0010189592838287353
3,100,570.0978035973023,0.0008886384963989257
4,100,563.8644516631573,0.0008687359094619752
5,100,578.8332319005056,0.0009161057472229005
6,100,578.0493029656659,0.0007898926734924316
7,100,541.1644260807742,0.0009311727115086148
8,100,586.5374578488988,0.0008757728338241576

```

  Throughput:

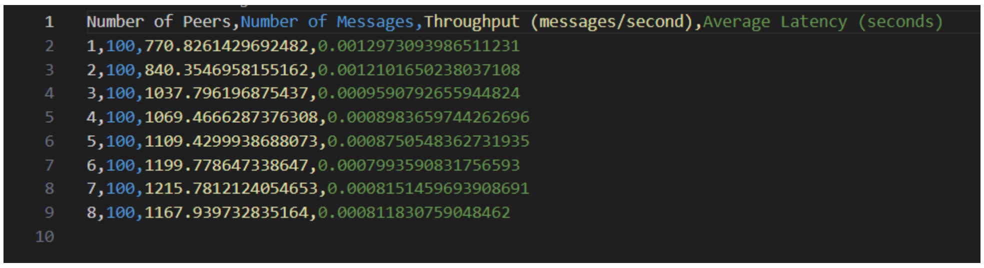


Latency:



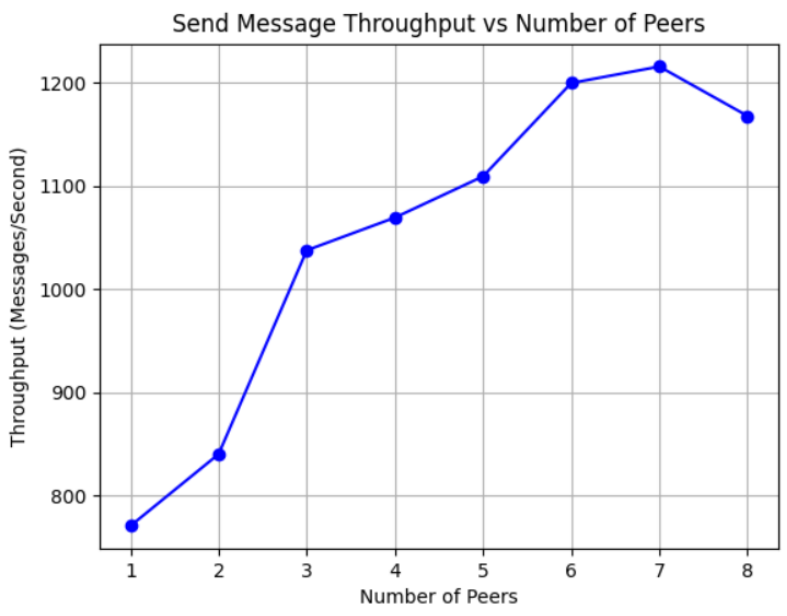
- **Send Message Benchmark:**

Output in CSV format:

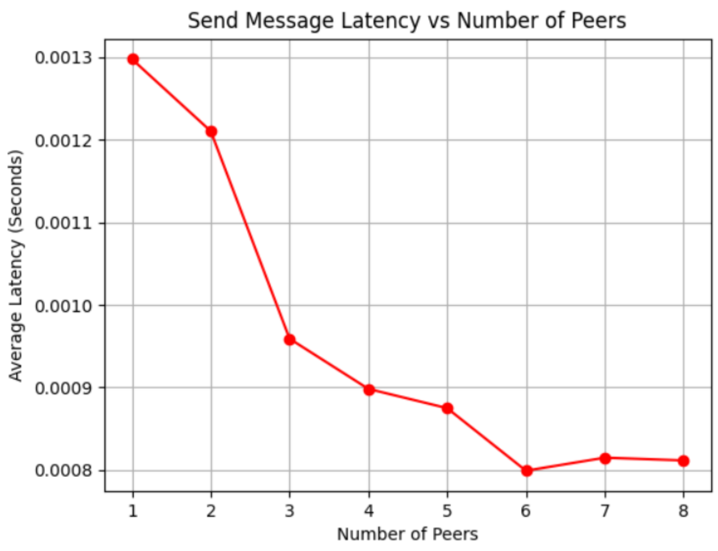


```
Number of Peers,Number of Messages,Throughput (messages/second),Average Latency (seconds)
1,100,770.8261429692482,0.0012973093986511231
2,100,840.3546958155162,0.0012101650238037108
3,100,1037.796196875437,0.0009590792655944824
4,100,1069.4666287376308,0.0008983659744262696
5,100,1109.4299938688073,0.0008750548362731935
6,100,1199.778647338647,0.0007993590831756593
7,100,1215.7812124054653,0.0008151459693908691
8,100,1167.939732835164,0.000811830759048462
```

Throughput:

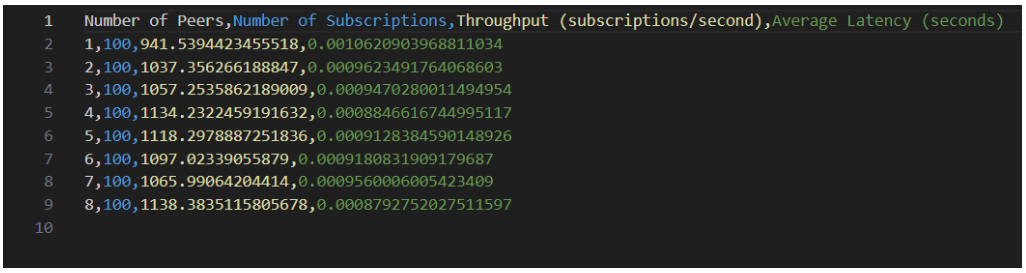


Latency:



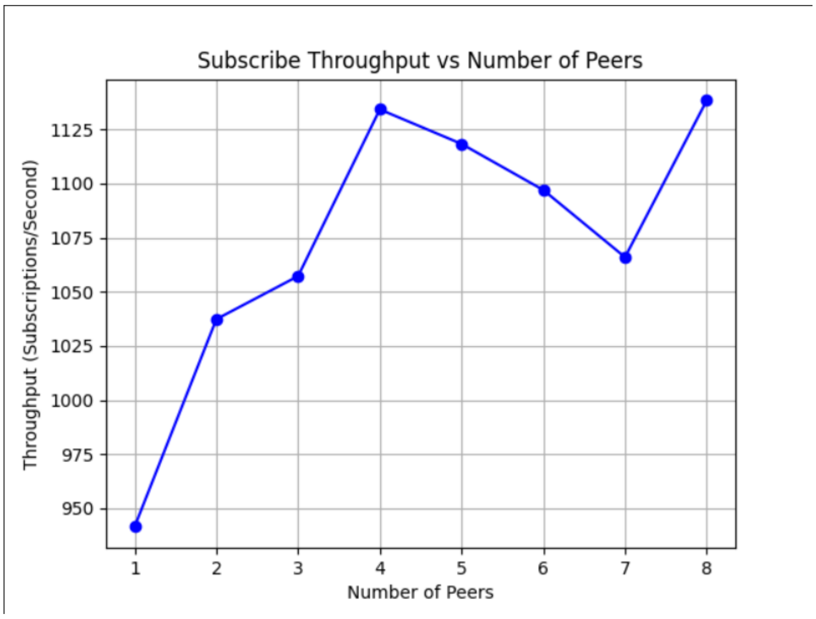
- **Subscribe Topic Benchmark:**

Output in CSV format:

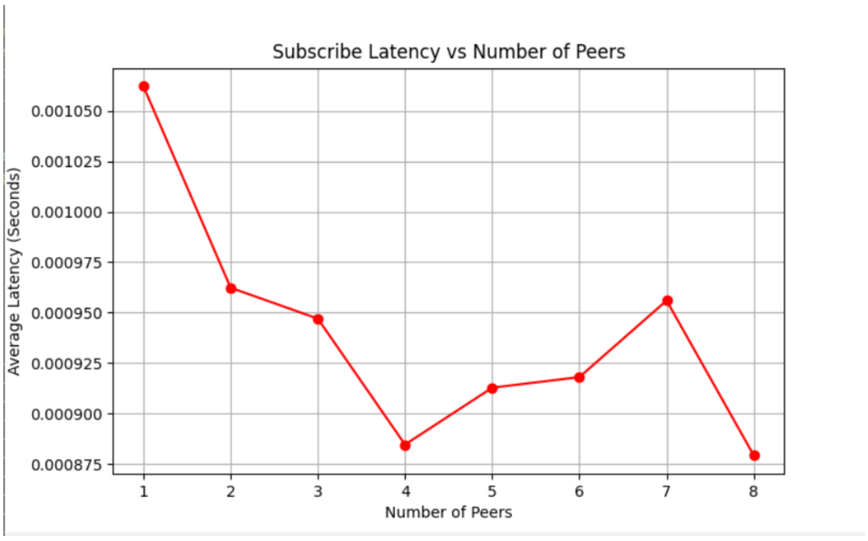


```
Number of Peers,Number of Subscriptions,Throughput (subscriptions/second),Average Latency (seconds)
1,100,941.5394423455518,0.0010620903968811034
2,100,1037.356266188847,0.0009623491764068603
3,100,1057.2535862189009,0.0009470280011494954
4,100,1134.2322459191632,0.0008846616744995117
5,100,1118.2978887251836,0.0009128384590148926
6,100,1097.02339055879,0.0009180831909179687
7,100,1065.99064204414,0.0009560006005423409
8,100,1138.3835115805678,0.0008792752027511597
```

Throughput:

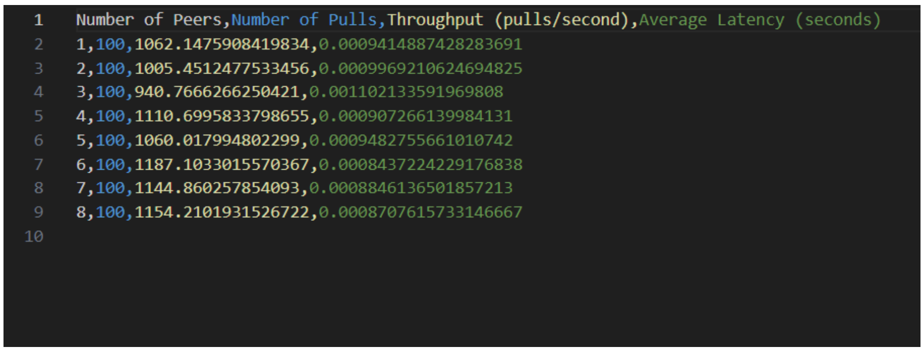


Latency:



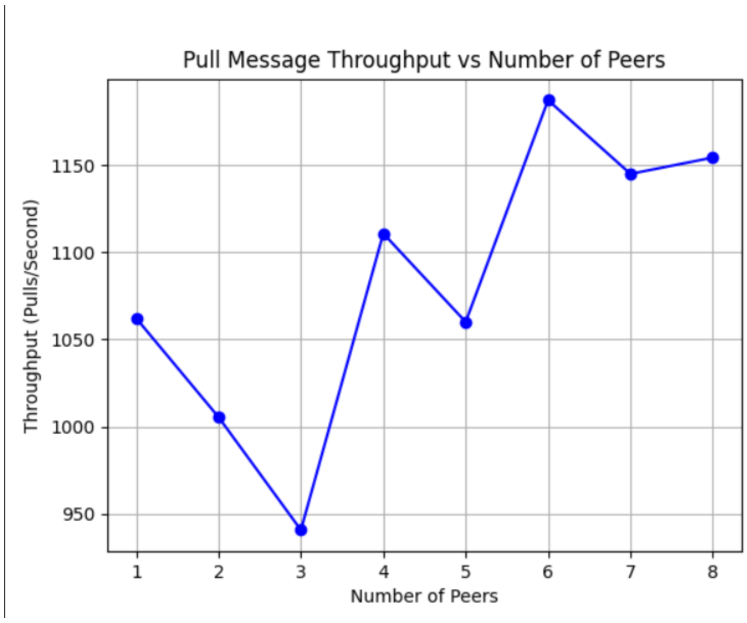
- **Pull Message Benchmark:**

Output in CSV format:

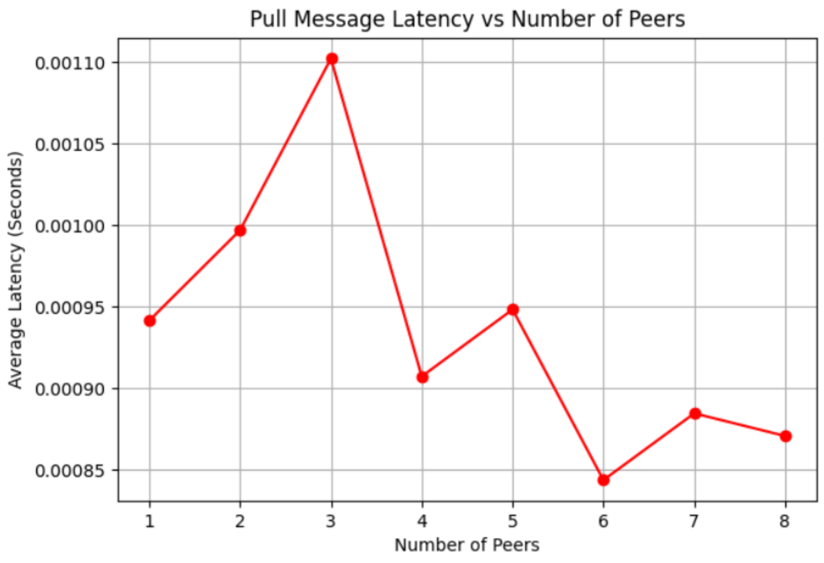


```
Number of Peers,Number of Pulls,Throughput (pulls/second),Average Latency (seconds)
1,100,1062.1475908419834,0.0009414887428283691
2,100,1005.4512477533456,0.0009969210624694825
3,100,940.7666266250421,0.001102133591969808
4,100,1110.6995833798655,0.000907266139984131
5,100,1060.017994802299,0.0009482755661010742
6,100,1187.1033015570367,0.0008437224229176838
7,100,1144.860257854093,0.0008846136501857213
8,100,1154.2101931526722,0.0008707615733146667
```

Throughput:



Latency:


Pull Message Latency vs Number of Peers
Average Latency (Seconds)
0.00110
0.00105
0.00100
0.00095
0.00090
0.00085
Number of Peers
1
2
3
4
5
6
7
8