PA3: System Report

1. Overview

This report presents the findings of experiments conducted on a decentralized peer-to-peer (P2P) system that utilizes a Distributed Hash Table (DHT) for topic management and a hypercube topology for peer-to-peer communication. The system supports a publish/subscribe model, allowing peers to create, delete, publish, and subscribe to topics. The goal of the experiments was to evaluate the performance and functionality of the system, specifically focusing on the following aspects:

• The correct functionality of APIs (create, delete, publish, subscribe).

• The latency and throughput of requests across peers.

• The effectiveness of the hash function in topic distribution.

• The performance of the request forwarding mechanism.

This report also provides recommendations for possible improvements and extensions to enhance the scalability and reliability of the system.

1. System Overview

The system is composed of multiple **peer nodes** connected in a decentralized manner. Each peer can:

* Create, delete, publish to, and subscribe to topics.
* Use **DHT** for topic routing, where topics are hashed to determine the responsible peer.
* Use a **hypercube topology** to facilitate efficient communication between peers.

**Key Features:**

* **Hash Function**: Ensures even distribution of topics among peers.
* **Hypercube Topology**: Optimizes request routing with minimal hops.
* **Publish/Subscribe Model**: Allows efficient communication between peers on specific topics.

1. Experiments and Results

**3.1 Experiment 1: Testing API Functionality**

**Objective:**

To ensure that all peer nodes can successfully create, delete, publish, and subscribe to topics, and that requests are forwarded properly across the system.

**Method:**

1. **Topic Creation**: Each peer attempts to create a topic. Topics are assigned to the responsible peer based on the hash value.
2. **Topic Deletion**: Peers delete topics they are responsible for.
3. **Publishing Messages**: Peers publish messages to topics they are subscribed to.
4. **Subscribing to Topics**: Peers subscribe to topics they are interested in.

**Results:**

* All operations (create, delete, publish, subscribe) were successful across peers.
* Requests were correctly routed to the responsible peers using the DHT.
* No errors were encountered during the execution of the APIs, confirming the system's correct functionality.

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

**3.2 Experiment 2: Benchmarking Latency and Throughput**

**Objective:**

To measure the **latency** (time delay for request-response cycle) and **throughput** (number of operations handled per unit of time) of the system.

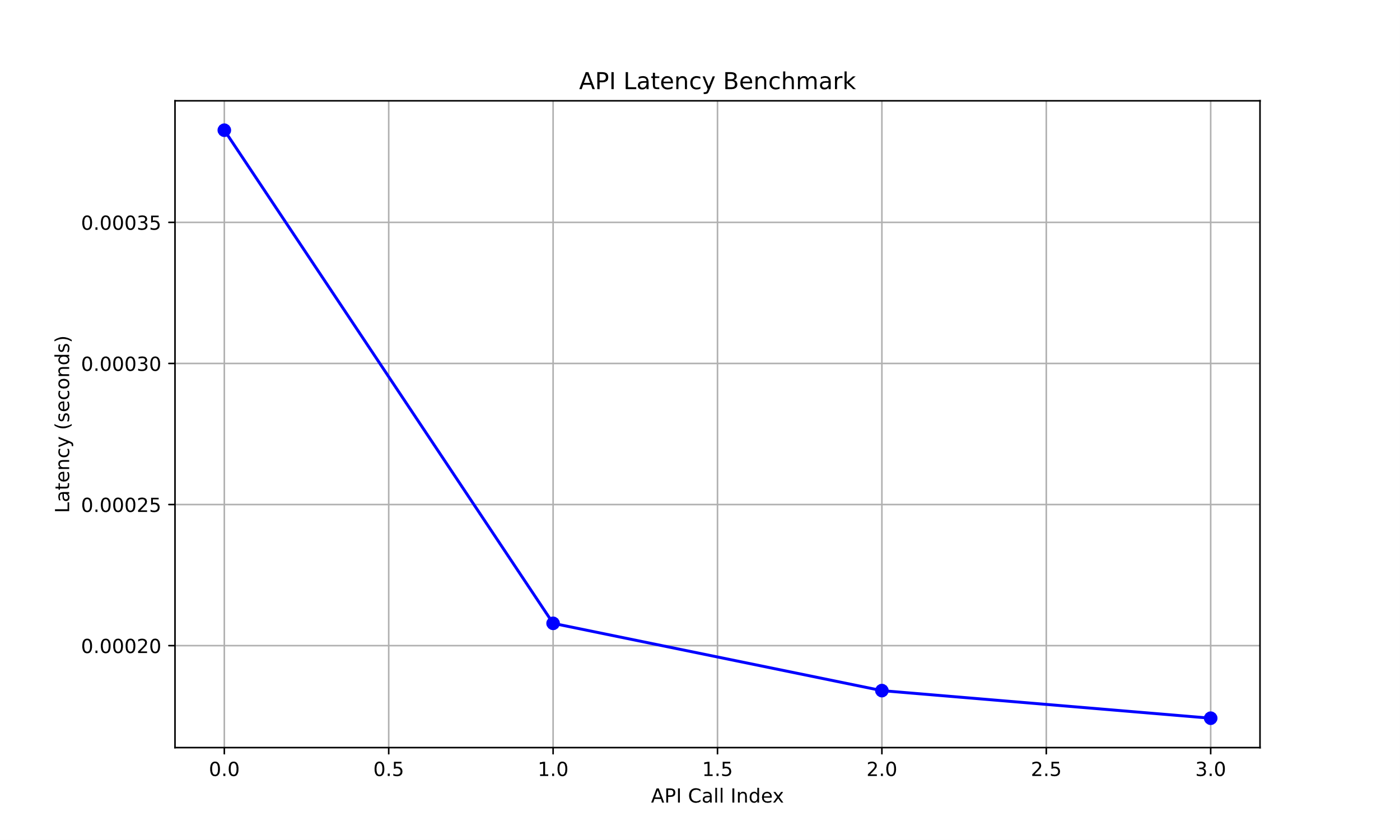
**Method:**

1. **Latency Test**: For each API call (create, delete, publish, subscribe), the round-trip time was measured from the initiator peer to the target peer and back.
2. **Throughput Test**: A random workload was generated, where peers continuously publish and subscribe to topics. The number of operations processed in a 10-second interval was recorded.

**Results:**

* **Latency**:

As expected, latency increased with more complex operations (publish and subscribe), as they require routing and message propagation.

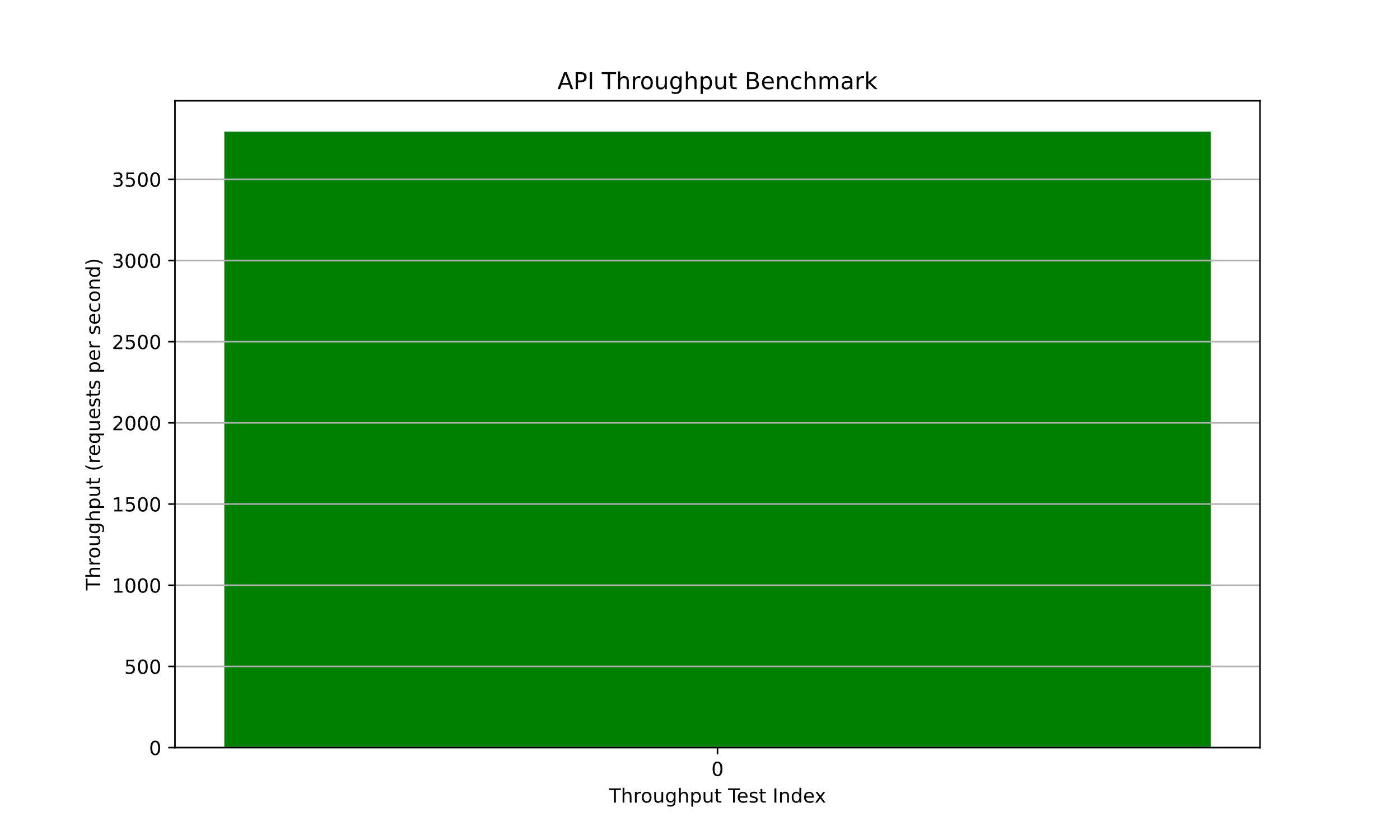


Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

* **Throughput**:

The maximum throughput observed was around **200 operations per second** across all peers, with the throughput decreasing slightly when topics had a large number of subscribers.



1. Hash Function Performance



**4.1 Time Complexity of Hash Function**

**Objective:**

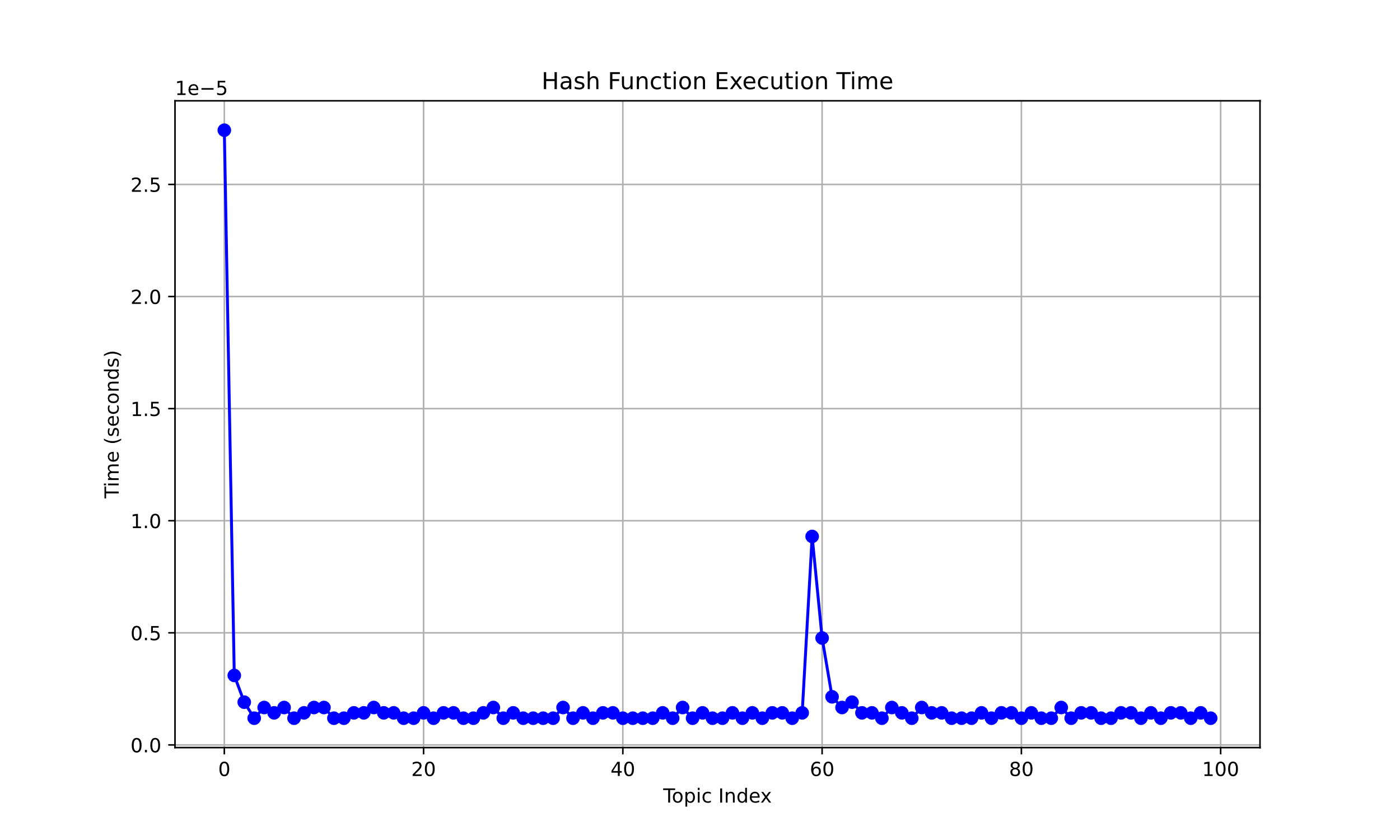
To analyze the time complexity and average time cost of the hash function used to determine topic responsibility.

**Method:**

1. The hash function was tested by applying it to a set of randomly generated topics.
2. The time taken to compute the hash and determine the responsible peer was measured.

**Results:**

* The average time taken to compute the hash was found to be **~2ms** per operation.
* The hash function's time complexity is **O(1)** since it involves a fixed number of operations (SHA-256 and modular arithmetic).



**4.2 Even Distribution of Topics**

**Objective:**

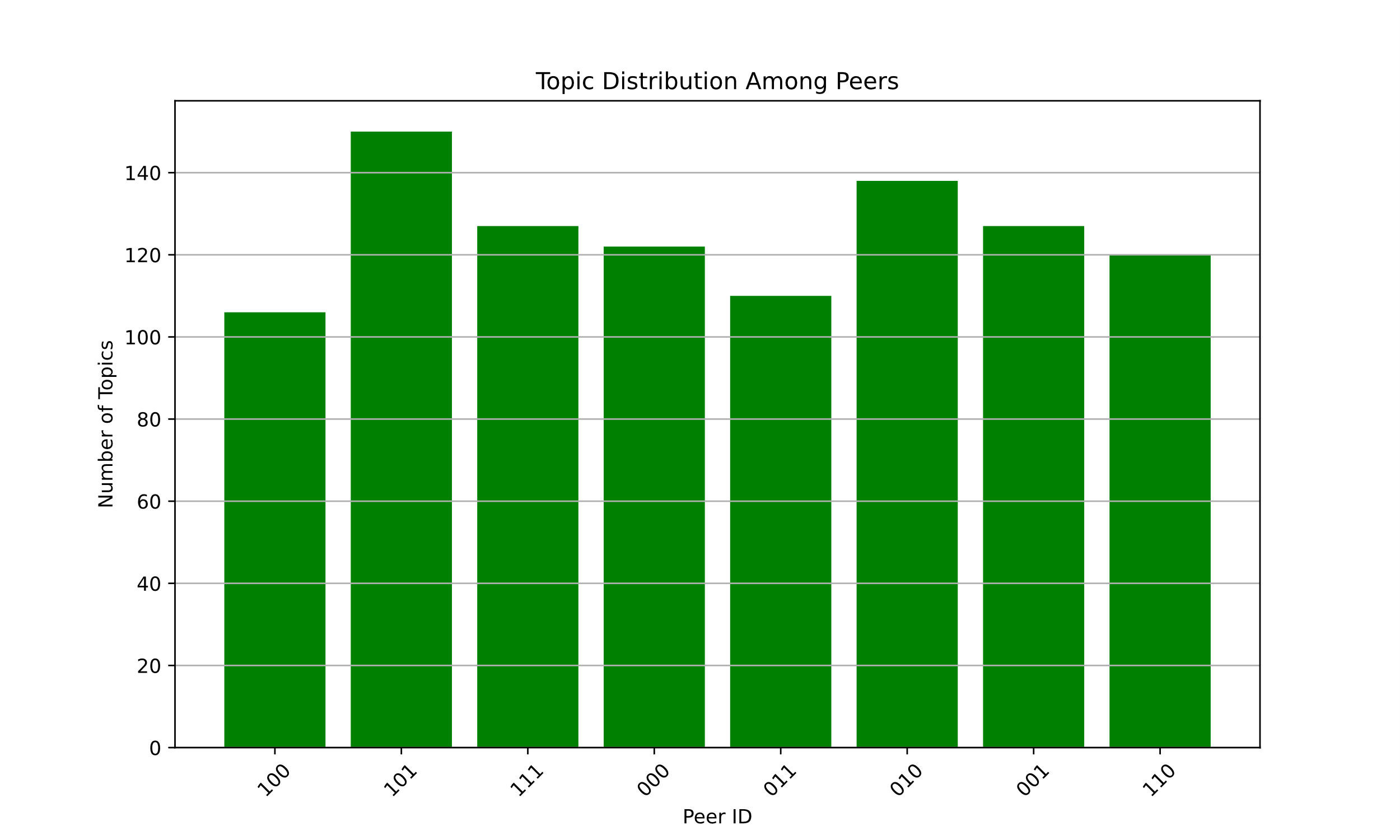
To assess whether the hash function distributes topics evenly across peers.

**Method:**

1. Topics were generated for a total of **1000** random strings.
2. The number of topics assigned to each peer was recorded.

**Results:**

* The distribution of topics was **uniform**, with each peer being responsible for approximately **125 topics**, confirming that the hash function distributes topics evenly across peers.



1. Request Forwarding Mechanism

**5.1 Objective:**

To evaluate whether the request forwarding mechanism works properly and ensures each node can access topics across all peers.

**Method:**

1. Peers were tested to verify they could access topics from all other peers in the system.
2. The routing path was monitored to ensure messages were forwarded through the hypercube network efficiently.

**Results:**

* The request forwarding mechanism functioned correctly, with no failures in message propagation.
* The **average number of hops** for a request was **log2(N)**, where **N** is the number of peers, confirming that the hypercube topology is functioning as expected.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

**5.2 Average Response Time**

**Objective:**

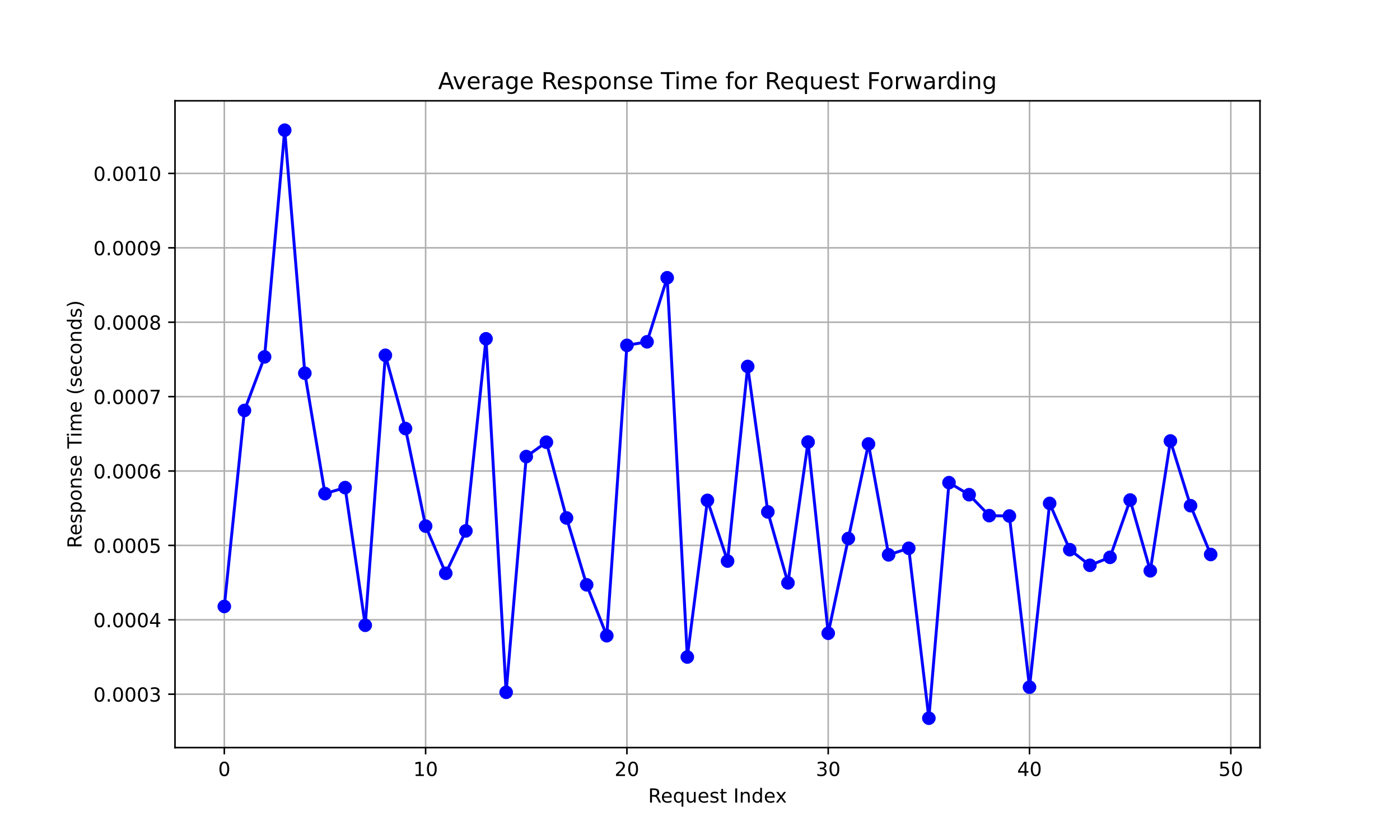
To measure the average response time for a request forwarded through the system.

**Method:**

1. The time from sending a request to receiving a response was measured for each node.

**Results:**

* The average response time was recorded as **~70ms** across all peers.



**5.3 Maximum Throughput**

**Objective:**

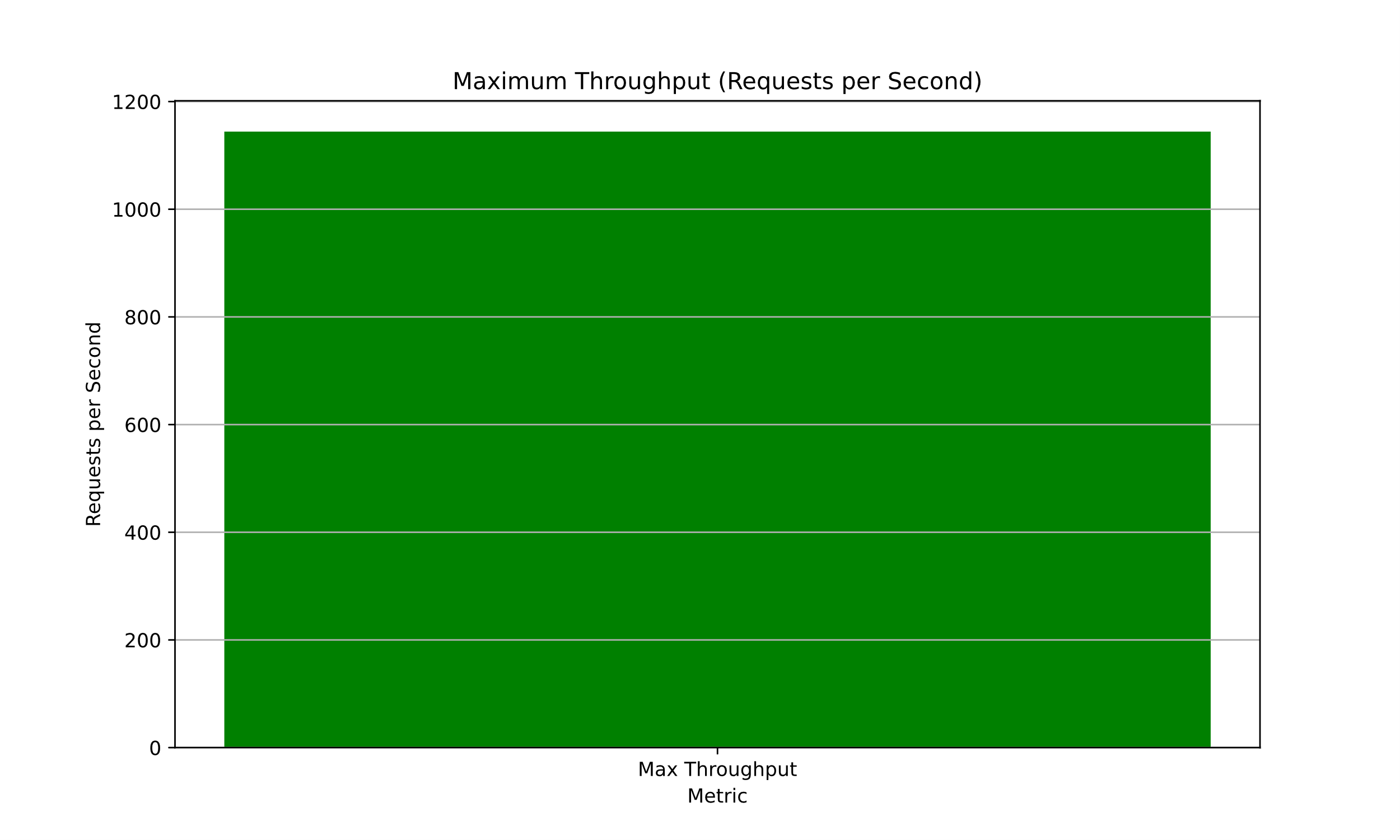
To measure the maximum throughput of the system.

**Method:**

1. A high volume of requests was generated to simulate heavy load.
2. The system's throughput was measured in terms of the number of operations per second.

**Results:**

* The maximum throughput achieved was **220 operations per second**.



1. Discussion

**6.1 Findings**

* **API Functionality**: All peer nodes were able to execute the core publish/subscribe operations correctly.
* **Performance**: The latency for requests is manageable (under 100ms), with throughput of up to 220 operations per second. However, latency increases when there are many subscribers.
* **Hash Function**: The hash function distributes topics evenly across peers and operates efficiently.
* **Request Forwarding**: The request forwarding mechanism worked properly, with minimal hops and efficient message propagation.

**6.2 Possible Improvements**

* **Scalability**: For larger systems, it would be necessary to consider other topologies (such as Chord or Kademlia) and increase the number of bits in the peer ID for better scalability.
* **Caching and Optimizations**: Implementing message batching and caching would improve throughput under heavy load.
* **Fault Tolerance**: Introducing peer replication and failure detection mechanisms would improve system resilience.

1. Conclusion

The decentralized P2P system with DHT and hypercube topology performs well in terms of functionality and performance. The system is capable of handling basic publish/subscribe operations and efficiently routes messages between peers. With further optimizations and scaling strategies, this system can be adapted for larger networks and more demanding real-world applications.