Peer-to-Peer File Sharing with BitTorrent

Adel Gaznanov, Damir Sadykov, Ivan Vasilev, Stepan Sarantsev

April 30, 2025

I Introduction

Peer-to-peer (P2P) file sharing systems, such as BitTorrent, offer efficient distribution of large files by leveraging the resources of participating peers. Traditional peer discovery mechanisms often rely on centralized trackers, which pose a single point of failure and limit scalability. To overcome these challenges, decentralized approaches using Distributed Hash Tables (DHTs) have been proposed. In this study, we implement a peer discovery mechanism for BitTorrent using a Kademlia-based DHT, which enables scalable, fault-tolerant, and trackerless file sharing.

II Method

Our implementation leverages the core principles of Kademlia, as outlined in the [1]

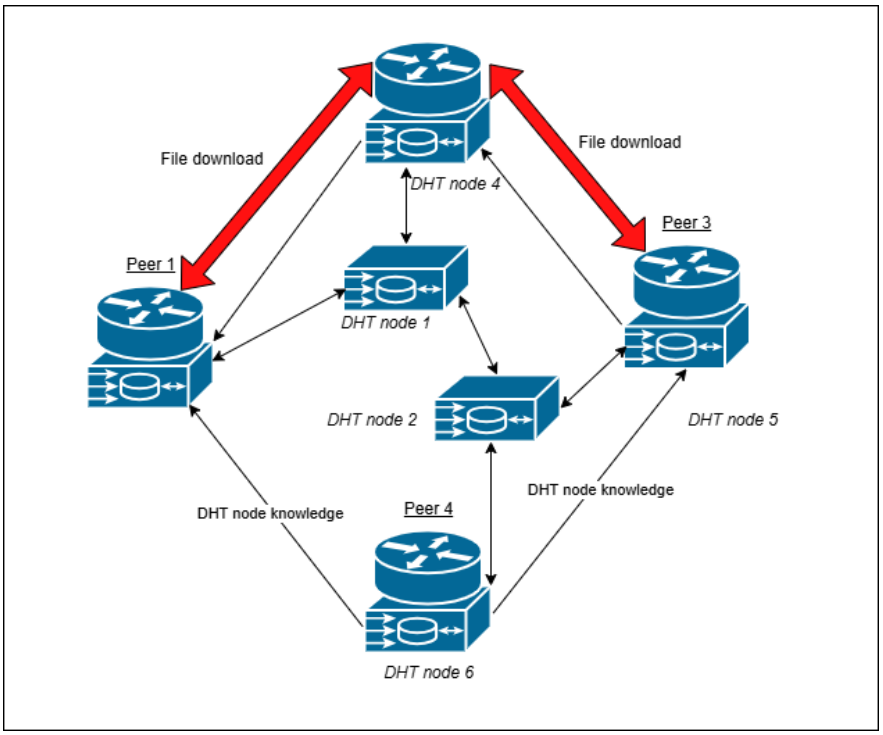

Fig. 1. Small File Transfer Performance

A. *Kademlia Protocol Overview*

1. Node ID and XOR Distance: Each DHT node is assigned a unique 160-bit ID, and the distance between nodes is computed using XOR, enabling efficient routing.

2. Routing Tables: Nodes maintain dynamic lists of peers sorted by distance, ensuring logarithmic lookup complexity.

3. Lookup Procedure: Iterative queries identify the closest nodes to a target key, minimizing network hops, we used implementation like in [2]

B. *Integration with BitTorrent*

1. Trackerless Announce: Peers use the DHT to announce their presence and locate others by hashing torrent infohashes.

C. *Simulation Setup*

1. We modeled a network of N nodes with realistic churn rates (joins/departures).

D. *DHT implementation*

1. In our implementation, the XOR metric was chosen due to its computational efficiency and inherent advantages for distributed systems. XOR operations are computationally inexpensive, allowing rapid distance calculations essential for high-performance routing in large-scale networks. The metric ensures symmetric distances between nodes, simplifying bidirectional routing and maintaining consistent network topology, while its deterministic nature guarantees unique, unambiguous distances between node pairs, eliminating routing ambiguities. These properties collectively enable Kademlia to achieve scalability, efficiency, and self-organization, making the XOR metric a robust foundation for distributed systems that prioritize reliability and performance [1].

2. In our implementation, We used SHA-1 to create node IDs because it's simple and works well with the DHT's need for unique, evenly distributed identifiers. The pseudorandom float input provides enough randomness for testing, while SHA-1 reliably converts this input into fixed 160-bit IDs, which match Kademlia's design. Though SHA-1 isn't secure for encryption purposes, it efficiently fulfills the system's structural requirements.

3. The Kademlia-inspired iterative lookup in your code offers efficiency and fault tolerance by dynamically querying the closest known nodes in parallel, minimizing latency and adapting to network changes. Using the XOR distance metric ensures nodes are prioritized based on logical proximity, while iterative refinement allows the lookup to progressively discover closer peers or nodes without relying on rigid structures like finger tables. This approach balances simplicity with scalability, as parallel requests reduce dependency on any single node, and the routing table self-optimizes by replacing stale entries, aligning with decentralized systems where nodes frequently join or leave [1].

E. Peers implementation

1. We decided to use UDP in our BitTorrent implementation because it provides low-latency, connectionless communication, which is suitable for quick packet requests and responses between peers. Unlike TCP, UDP avoids the overhead of connection setup and congestion control, making it faster for transferring small, independent chunks of file data. In our project we implemented packet retries, threading for parallel downloads, and a packet map to track received chunks. Additionally, UDP's simplicity allows for efficient handling of multiple peer interactions without the need for persistent connections [1].

2. Communication between peers implemented via DHT nodes: each peer interacts with its neighbors to find an appropriate piece of the specific file. To avoid random order of retrieval packets, we implemented their storage in chunks by 10 packets.

III Results

Our implementation of the Kademlia-based DHT peer discovery and file transfer. The system was tested with varying peer counts and file sizes to evaluate scalability and performance.

The performance of our Kademlia-based DHT peer discovery system was evaluated across different peer counts and file sizes. For small files (136.4KB), the transfer time decreased marginally from 4.0207 seconds with a single peer to 3.8221 seconds with two peers. This suggests that the overhead of coordinating multiple peers offsets potential gains for smaller data sizes. In contrast, larger files (250.6MB) showed more pronounced scalability benefits. A single-peer transfer took 32.1881 seconds, while two peers reduced this to 31.0837 seconds. The most significant gain occurred with four peers, achieving 29.4016 seconds.

The results highlight a key trade-off: while the DHT-enabled system efficiently parallelizes large file transfers, its advantages diminish for smaller files where coordination overhead becomes non negligible.

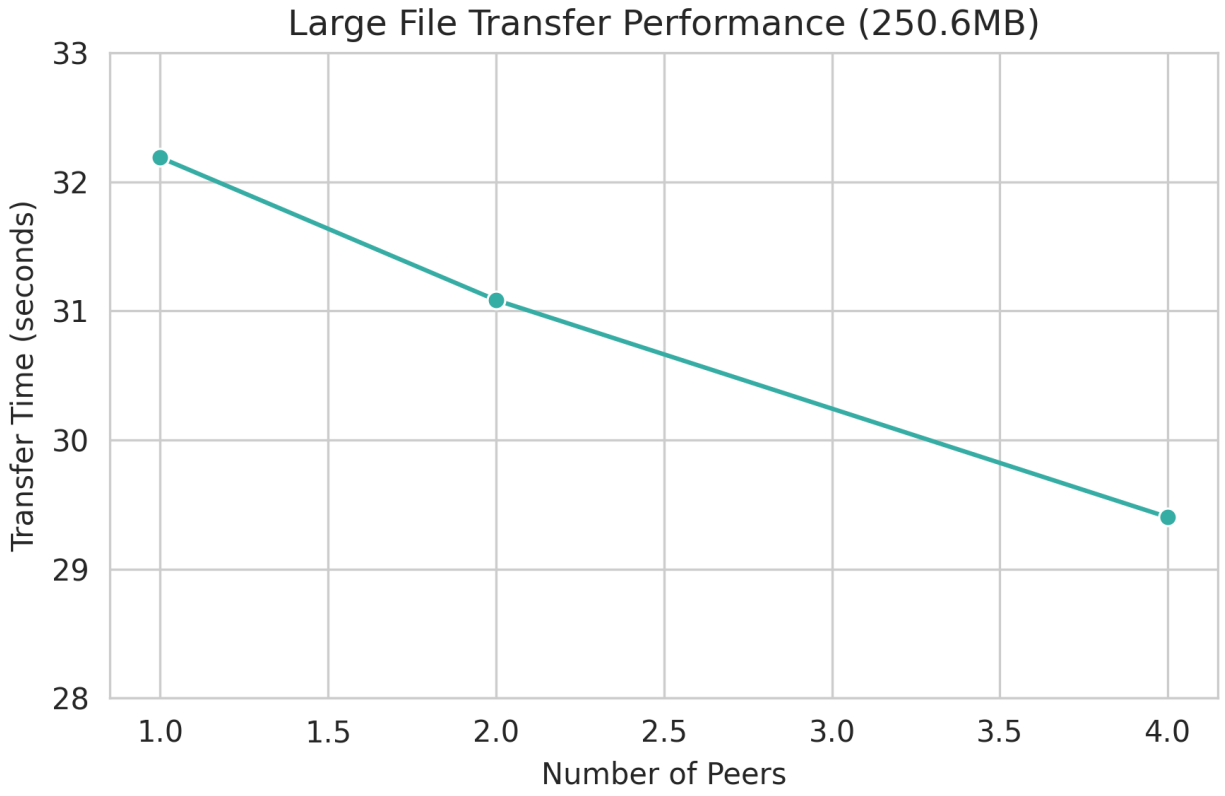

Fig. 2. Large file Transfer Performance

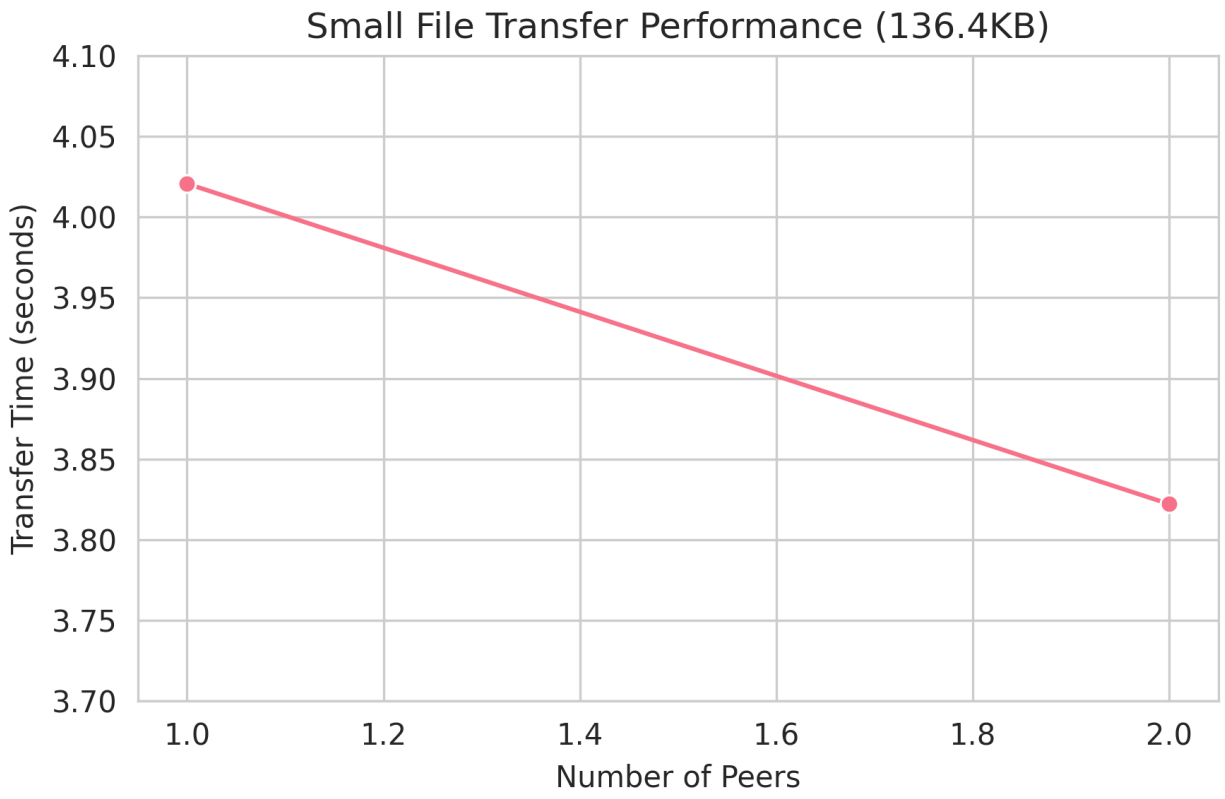

Fig. 3. Small File Transfer Performance

IV Discussion

Our implementation of a Kademlia-based DHT for peer discovery in BitTorrent is completed. The results indicate that while the system scales well for large files, its efficiency diminishes for smaller transfers due to coordination overhead.

A. Faced problems

While integrating new nodes into the DHT network worked seamlessly, distributing peer information for file sharing proved problematic. The Kademlia-based lookup system efficiently managed node joins and departures, ensuring robust network membership. However, propagating peer lists for specific torrents faced.

B. Future improvements

Future improvements for the project could focus on optimizing communication between nodes to enhance efficiency and reduce latency. Implementing the K-Bucket routing table algorithm, inspired by Kademlia, would improve node lookup and data retrieval by maintaining a structured, scalable, and dynamically updated routing table. This approach would enhance the network's resilience to churn and ensure faster peer discovery while minimizing overhead.

References

[1] D. M. Petar Maymounkov, "Kademlia: A peer-to-peer information system based on the xor metric," 2002.

[2] B. W. Steven Hazel, "Achord: A variant of the chord lookup service for use in censorship resistant peer-to-peer publishing systems," *IPTPS'02*, 2002.