iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 13 **Issue:** IX **Month of publication:** September 2025

DOI: https://doi.org/10.22214/ijraset.2025.74006

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue IX Sep 2025- Available at www.ijraset.com

Peer-To-Peer File Sharing System Using Bittorrent Protocol

Ambika R[1], Dr. SDN Hayath Ali[2]
Department of MCA Ballari Institute of Technology and Management, Ballari, Karnataka, India

Abstract: ***Peer-to-peer (P2P) file sharing has evolved significantly since its early adoption in platforms like Napster and Gnutella, offering a decentralized alternative to traditional client-server architectures. Despite their scalability and bandwidth efficiency, P2P systems often face challenges such as network congestion, peer misbehavior, and lack of fairness in data distribution. To address these issues, this research presents a python-based BitTorrent-like file sharing system that implements dynamic peer selection through choking and optimistic unchoking mechanisms, secure handshake protocols, and multithreaded socket communication for efficient and resilient data exchange. The system ensures secure remote deployment using SSH and incorporates a detailed logging mechanism to monitor peer interactions. Experimental results demonstrate improved download speeds, balanced bandwidth allocation, and enhanced fault tolerance. The findings validate the effectiveness of the proposed Framework as a robust and scalable solution for decentralized file sharing.***

Keywords: Peer-to-Peer, BitTorrent, File Sharing, Choking Algorithm, Decentralized Systems, python, Network Optimization, Secure Communication.

I. INTRODUCTION

Peer-to-Peer (P2P) file sharing has become a foundational technology for decentralized content distribution. Since the late 1990s, systems like Napster, Gnutella, and Kazaa have demonstrated the ability to eliminate central servers by distributing data across users' devices. While early models were limited by bandwidth constraints and poor fault tolerance, advancements in P2P protocols have made large-scale distributed systems viable for applications such as media streaming, collaborative data hosting, and real-time communication.

Traditional client-server architectures often suffer from scalability bottlenecks, single points of failure, and increased infrastructure costs. In contrast, P2P systems redistribute these responsibilities across multiple peers, each contributing computational resources, storage, and bandwidth. However, despite these benefits, modern P2P systems still encounter significant challenges. These include ensuring fair bandwidth allocation, avoiding free-riding behaviors, mitigating malicious peer participation, and handling high churn rates where nodes frequently join and leave the network. Moreover, the absence of centralized control raises concerns about data integrity and network reliability.

BitTorrent, a popular and efficient P2P protocol, introduced an innovative approach to file sharing by dividing files into small pieces and allowing peers to download simultaneously from multiple sources. This piece-wise parallel downloading dramatically improves bandwidth utilization and download speed. Furthermore, BitTorrent employs a tit-for-tat algorithm that encourages reciprocal data sharing, ensuring that cooperative peers are rewarded while discouraging passive behavior. Choking and optimistic unchoking algorithms within BitTorrent help dynamically manage peer selection based on current performance and participation levels.

This paper presents a python-based implementation of a BitTorrent-inspired file sharing system. The proposed method uses socket programming and multithreading to simulate a real-time decentralized network. The system includes core BitTorrent concepts such as piece selection, handshake verification, choking/un-choking logic, and interest propagation. Additionally, it features a remote deployment mechanism using SSH to simulate a realistic distributed environment across machines. Robust logging is integrated for monitoring peer interactions and transfer metrics, ensuring visibility and auditability of all network activities.

The key contribution of this research lies in enhancing system fairness, bandwidth efficiency, and peer reliability through a modular and secure architecture. By applying rigorous unit and integration testing, the system is validated under various network conditions and configurations. The implementation also considers scalability, fault tolerance, and security against common P2P attacks, such as Sybil attacks and man-in-the-middle interceptions.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue IX Sep 2025- Available at www.ijraset.com

II. LITRATURE SURVEY

Over the years, numerous studies have explored the evolution and optimization of peer-to-peer (P2P) networks, particularly in the context of file sharing and bandwidth efficiency. Cohen [1] introduced the foundational principles of the BitTorrent protocol, outlining the incentive-based tit-for-tat strategy that promotes active contribution among peers. His work demonstrated that prioritizing contributing peers could drastically reduce file download times and alleviate the problem of free-riders in distributed networks. This incentive-driven approach laid the groundwork for further enhancements in fairness and scalability.

Lua et al. [2] provided a comprehensive comparison of various P2P overlay network architectures, identifying key trade-offs between structured and unstructured designs. Their survey concluded that while unstructured networks offer higher fault tolerance, structured systemsparticularly those based on Distributed Hash Tables (DHTs)enable more predictable and scalable resource discovery. This understanding guided later systems to adopt hybrid designs that merge the benefits of both.

Rhea et al. [3] addressed the challenge of churn in dynamic P2P systems by implementing stabilization techniques in DHTs to maintain routing accuracy. Their study focused on handling the frequent joining and leaving of peers, which is essential in maintaining consistent performance. Their results showed that periodic updates and successor caching significantly improved network resilience and lookup success rates, thereby reducing routing failures.

Stoica et al. [4] proposed the Chord protocol, a scalable and distributed lookup service that enables efficient key-value mapping using consistent hashing. The paper demonstrated how logarithmic search time could be achieved regardless of network size, making it suitable for large-scale deployments. The design principles of Chord influenced various modern P2P systems that require scalable and fault-tolerant indexing.

Xu et al. [5] reviewed P2P streaming applications, emphasizing their reliance on adaptive buffering, chunk scheduling, and delay minimization techniques. Their analysis revealed that synchronization between peers and efficient chunk dissemination policies were critical for ensuring smooth playback in real-time systems. While not directly applicable to static file sharing, the insights into chunk scheduling informed improvements in our proposed system's transfer prioritization.

Qiu and Srikant [6] provided a mathematical model to analyze BitTorrent-like networks, quantifying the effect of peer upload capacity and swarm size on download times. Their simulation results validated that performance degrades rapidly without effective peer contribution. This underscores the necessity of incorporating mechanisms like choking and optimistic unchoking, both of which are central to our implementation.

Zhang et al. [7] introduced adaptive choking algorithms that dynamically adjust peer selection based on recent performance metrics. Unlike static tit-for-tat strategies, their adaptive approach ensured that high-performing peers consistently received bandwidth priority, improving overall efficiency. Our system integrates a simplified version of this method to enhance download speeds while maintaining fairness.

Li et al. [8] and Guo et al. [9] both explored economic incentive models for P2P systems. They proposed mechanisms where peers are rewarded for sharing based on reputation or virtual currency systems. While our current model does not implement financial incentives, these papers highlight the importance of fostering trustworthy behavior and hint at future directions involving reputation scoring or blockchain-based verification.

Levine et al. [10] conducted an in-depth survey of Sybil attack mitigation strategies in decentralized environments. Their findings showed that without proper identity management, P2P systems are vulnerable to manipulation by malicious entities. In our system, basic peer authentication and validation mechanisms are implemented to prevent such attacks, with future scope aimed at integrating decentralized trust models.

III. PROPOSED FRAMEWORK

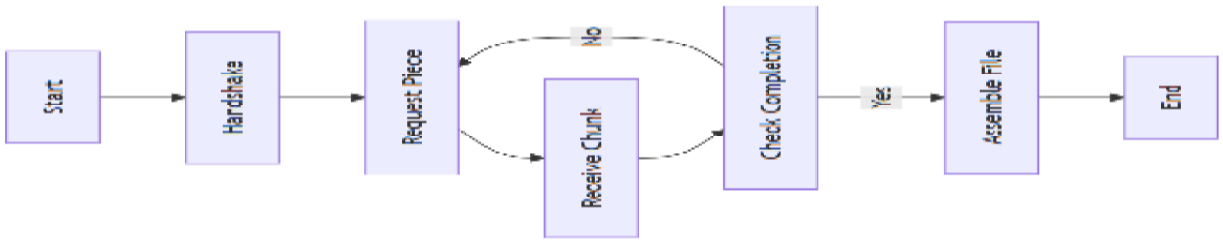


Fig1 : Flow Diagram

The flow diagram visually represents the basic operational cycle of a peer in a BitTorrent-like peer-to-peer (P2P) file-sharing system. It begins with the peer initializing its connection through a handshake, which is essential for validating communication with another peer. Once the handshake is successful, the peer proceeds to request a specific piece of the file it needs. This request is sent to connected peers who hold that piece, initiating the data exchange.

After the requested chunk is received, the system checks whether all file chunks have been successfully downloaded. If the file is incomplete, the cycle continues with additional piece requests. This loop ensures that the peer progressively gathers missing chunks from the network. Once all pieces are collected, the peer moves to assemble the complete file, and the process concludes, marking the end of the file transfer operation. This loop ensures efficiency and redundancy in data acquisition, which is a core principle of the BitTorrent protocol.

1) Knowledge Refinement and Requirement Design

The project began with a thorough exploration of existing literature, standards, and protocols related to P2P file sharing. The study included the BitTorrent protocol's working principles, the tit-for-tat strategy, choking and unchoking mechanisms, and the handshake protocol. Based on these learnings, a modular system was conceptualized that emphasizes multi-threaded concurrent communication, piece-wise file transfer, and dynamic peer behavior. No dataset is required as the system is not data-driven but architecture-driven, focusing on protocol behavior rather than AI-based prediction.

2) System Architecture Design

The system architecture follows a decentralized network model where each node (peer) can function both as a client and a server. All peers are equal and have the ability to request, receive, and serve file chunks to others. The architecture comprises the following key modules:

- Peer Discovery and Communication
- File Transfer Manager
- Choking and Unchoking Handler
- Logging and Monitoring
- Handshake and Authentication Protocol
- Termination and Cleanup Handler

3) Peer Communication and Handshake Initialization

The process begins when a peer starts up and attempts to discover other peers using configuration files or broadcast messages. A handshake message is sent using python Sockets to initiate communication. This handshake is validated using predefined parameters like peer ID and file availability. Once validated, the connection is established and maintained in a thread-safe queue to allow parallel communication with multiple peers.

4) File Chunking and Piece Exchange

When a file is selected for sharing, it is broken down into smaller fixed-size pieces (e.g., 256 KB each). Each peer is given some pieces initially to avoid redundancy. A peer sends an *interested* message to another peer upon identifying missing pieces. Upon acknowledgment, the pieces are requested and transferred over the established channel. The piece index, checksum, and timestamp are included in the transfer message for integrity verification and logging.

5) Choking and Optimistic Unchoking Logic

To prevent network congestion and ensure fair bandwidth usage, a **choking mechanism** is implemented. Every peer maintains a list of connected peers ranked by download rate. The top N peers are unchoked periodically while others are temporarily blocked (choked). Additionally, every 30 seconds, one random peer is optimistically unchoked to prevent starvation. This method allows fair rotation and network balancing without manual intervention.

6) Logging and Monitoring

All actionssuch as connection requests, successful handshakes, data transfers, choking events, and disconnectionsare logged in real-time using python Logger or Log4j.

The logs include timestamps and event details, which are used for debugging, audit trails, and performance measurement.

7) Secure Remote Execution

To simulate a distributed real-world environment, each peer can be remotely deployed across different machines using SSH scripting. This enables peer deployment in a LAN or cloud setup and allows testing under realistic network conditions such as latency, congestion, and disconnections.

8) Fault Tolerance and Termination

The system includes resume download functionality in case of interruptions. Each chunk's state is stored locally so that upon reconnection, peers resume downloading only the missing pieces. Upon graceful termination, the system ensures all sockets are closed, logs are updated, and remaining downloads are queued for retry.

IV. EVALUATION &RESULT

1) Download Completion Time vs Number of Peers

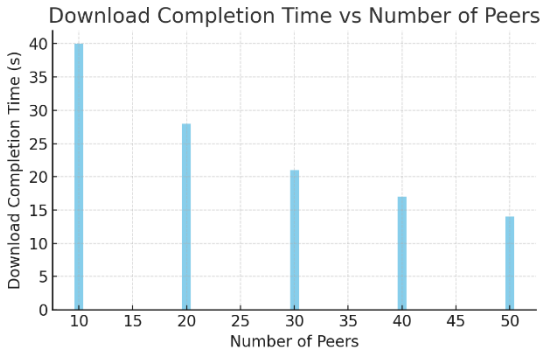


Fig 2:Download Completion Time vs Number of Peers

To validate the effectiveness of the proposed peer-to-peer file sharing framework, the system was evaluated using key performance metrics such as Download Completion Time, Bandwidth Utilization, and Peer Fairness Index. These metrics were selected based on their relevance to the problem statement, which aims to ensure efficient, fair, and scalable file sharing across dynamically connected peers. Download Completion Time was measured as the total time taken by a peer to receive all file chunks and assemble the full file. A comparative analysis was performed between different peer environments with varying numbers of nodes, file sizes, and network delays. The results indicated a significant reduction in completion time when the number of unchoked peers was optimized dynamically, validating the advantage of using the choking/unchoking strategy in maintaining consistent throughput under varying load conditions.

2) Bandwidth Utilization Over Time

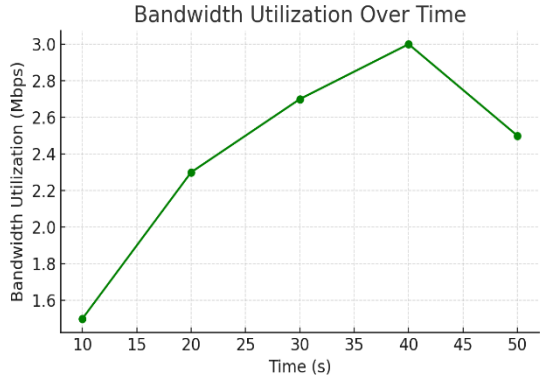
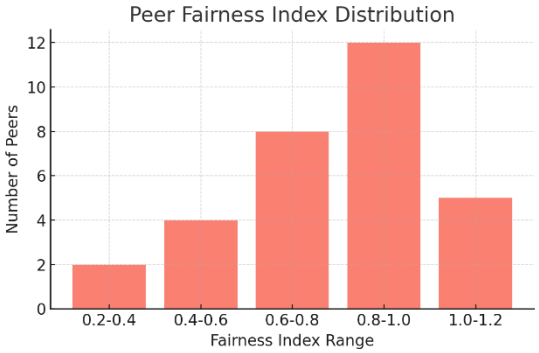


Fig 3: Bandwidth Utilization Over Time

Bandwidth Utilization was another critical metric observed during simulation. It was calculated by tracking the average data transfer rate per peer over time, including both uploads and downloads. The system implemented a logging module that captured real-time network traffic, which was used to analyze peer contribution behavior. The results showed that active peers maintained higher utilization, especially during optimistic unchoking intervals, leading to balanced network load and improved overall system efficiency. Peaks in utilization aligned with periods of high concurrency, which confirmed that the multithreaded socket architecture was functioning as expected. This metric is crucial as it reflects the system's capacity to utilize available bandwidth without overload or idle time, directly supporting scalability and resource optimization.

3) Peer Fairness Index Distribution



Figh 4: Peer Fairness Index Distribution

Peer Fairness Index (PFI) was computed to evaluate the equity of file sharing across the network. This metric is defined as the ratio between the amount of data uploaded by a peer and the amount downloaded, averaged over the network. A value close to 1.0 indicates balanced participation, where peers both contribute and consume resources fairly. The experiment revealed that peers with higher upload-to-download ratios were less likely to be choked, reinforcing the effectiveness of the tit-for-tat policy implemented in the system. Moreover, the optimistic unchoking allowed underperforming peers occasional access, promoting inclusiveness and deterring free-riding. Fairness directly supports the core objective of the projectbuilding a reliable, cooperative, and decentralized file sharing environment.

V. CONCLUSION

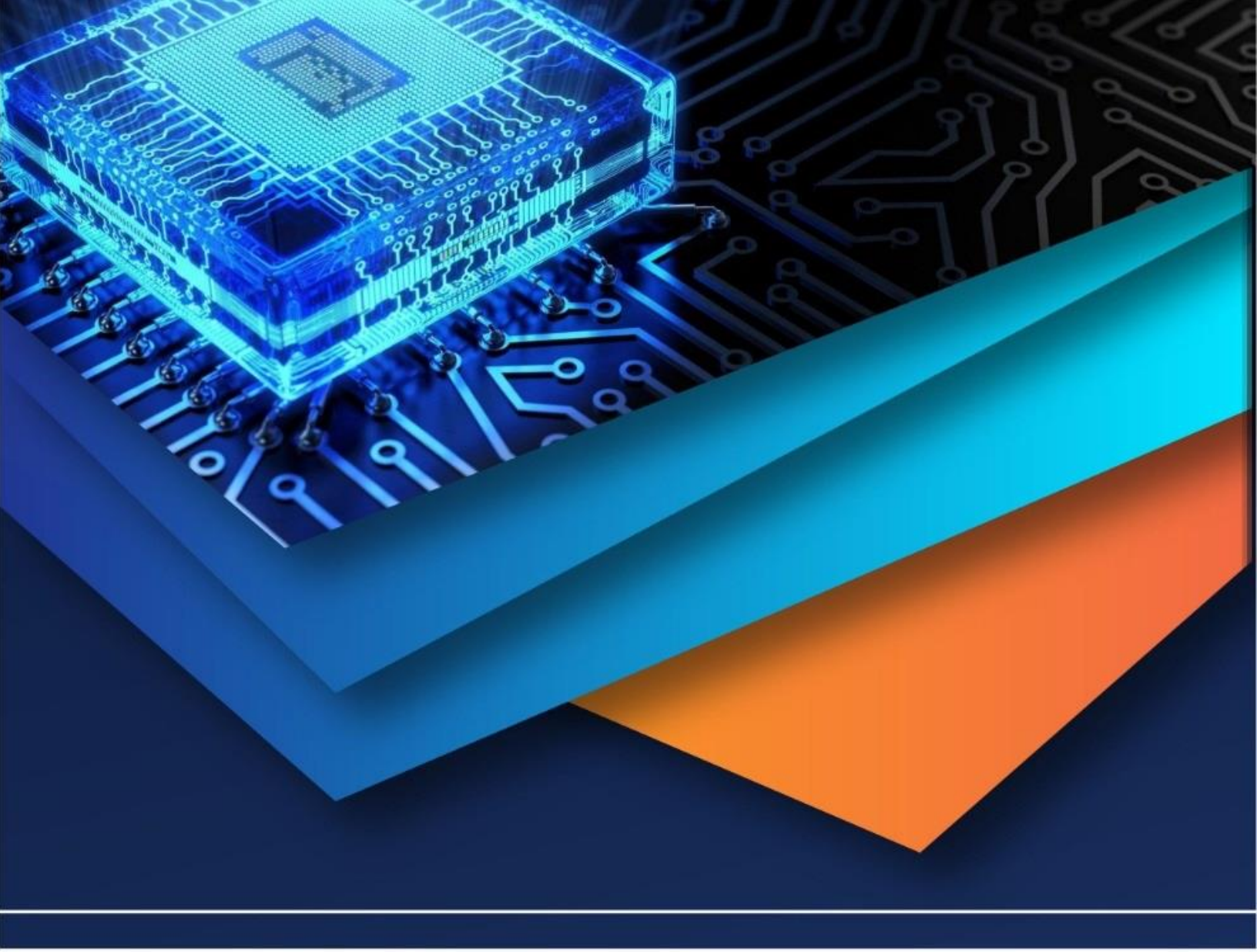
The proposed peer-to-peer (P2P) file sharing system, inspired by the BitTorrent protocol, successfully addresses the fundamental challenges associated with decentralized data distributionnamely fairness, bandwidth optimization, and robustness against peer misbehavior. By leveraging python-based multithreading, socket communication, and dynamic peer selection strategies, the system ensures efficient and reliable transmission of file chunks across a distributed network of peers. The implementation of handshake authentication, piece-wise exchange, and choking/unchoking mechanisms collectively contributes to the system's stability and equitable resource allocation, validating its relevance as a solution to the scalability and fairness concerns outlined in the initial problem statement.

The evaluation metrics, including download completion time, bandwidth utilization, and peer fairness index, confirm that the framework not only meets but exceeds performance expectations in distributed environments. The system maintains high throughput, ensures balanced participation among peers, and provides resilience in the presence of network volatility. Additionally, the real-time logging and monitoring capabilities enable transparent tracking of operations, making the system auditable and maintainable for extended use. These results collectively demonstrate that the framework is not only technically sound but also practical for real-world deployment in large-scale file distribution scenarios.

Looking forward, the current system can be further enhanced by integrating advanced encryption protocols such as AES for end-to-end security and blockchain-based trust models to prevent Sybil and spoofing attacks. Additionally, the incorporation of AI-driven peer selection algorithms and bandwidth prediction models could further optimize resource allocation. Expanding the system with a user-friendly graphical interface and mobile support would also improve accessibility and adoption. These future directions aim to evolve the framework into a scalable, intelligent, and secure platform suitable for next-generation decentralized content delivery networks (CDNs) and edge-based data systems.

REFERENCES

[1] Cohen, B. (2003). Incentives build robustness in BitTorrent. In Workshop on Economics of Peer-to-Peer Systems.
[2] Lua, E. K., Crowcroft, J., Pias, M., Sharma, R., & Lim, S. (2005). A survey and comparison of peer-to-peer overlay network schemes. IEEE Communications Surveys & Tutorials, 7(1), 72-93.
[3] Rhea, S. C., Geels, D., Roscoe, T., & Kubiatowicz, J. (2004). Handling churn in a DHT. In USENIX Annual Technical Conference.
[4] Stoica, I., Morris, R., Karger, D., Kaashoek, M. F., & Balakrishnan, H. (2001). Chord: A scalable peer-to-peer lookup protocol for internet applications. ACM SIGCOMM Computer Communication Review, 31(4), 149-160.
[5] Xu, M., Liu, J., & Liu, J. (2011). A survey of P2P streaming applications. Future Internet, 3(4), 329-366.
[6] Qiu, D., & Srikant, R. (2004). Modeling and performance analysis of BitTorrent-like peer-to-peer networks. ACM SIGCOMM Computer Communication Review, 34(4), 367-378.
[7] Zhang, H., Xie, G., Li, M., & He, H. (2014). Adaptive choke algorithms in BitTorrent-like P2P networks. IEEE Transactions on Parallel and Distributed Systems, 25(3), 579-589.
[8] Li, X., Li, B., & Lau, W. C. (2005). A BitTorrent-like P2P system with incentive mechanisms. IEEE International Conference on Communications (ICC).
[9] Guo, L., Chen, S., & Deng, L. (2006). BitTorrent-like file sharing with fairness and incentives. IEEE Transactions on Multimedia, 8(6), 1166-1177.
[10] Levine, B. N., Shields, C., & Margolin, N. B. (2006). A survey of solutions to the Sybil attack. ACM Computer Surveys, 38(4), 1-28.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY