Journal of Data Acquisition and Processing
ISSN: 1004-9037
https://sjcjycl.cn/
DOI: 10.5281/zenodo.7778222

EFFICIENT PEER TO PEER LIVE STREAMING MODEL USING CONNECTION SWITCHING TO IMPROVE THE QUALITY OF SERVICE

M. Satyanarayana Reddy[1] and P. Chenna Reddy[2]
[1]Research Scholar, Department of CSE, JNTUA Anantapuramu, Andhra Pradesh, India
[2]Professor, Department of CSE, JNTUA Anantapuramu, Andhra Pradesh, India
Mail id: satyam.marri@gmail.com, chennareddy.cse@jntua.ac.in

Abstract: Peer-to-Peer (P2P) methods have been regarded as an ideal solution for large live streaming platforms due to their low cost and scalability, they tend to cause delays in playback. In order to address this issue, we have proposed a method that allows a P2P system to select its neighbour peers more efficiently. The proposed method allows the system to select its neighbour peers even if the number of connectable peers has already exceeded the maximum. This method would greatly increase the data duplication among the peers and make the system's playback range denser. To prevent degradation in the quality of the system's playback, the method only performs connection switching if all of the other connectable peers have sufficient buffer data. Through extensive simulations, the proposed method significantly lowers the start-up latency and playback lag.
Keywords: P2P live Streaming, Playback, connection Switching, Network establishment

1. Introduction

The high-speed broadband network that has allowed communication and broadcasting technologies to converge has been credited for the emergence of IPTV services. While most IPTV systems adopt a single server/client architecture, providing services to a large number of users necessitates the use of a content distribution network. As the number of users increases, the cost of maintaining network capacity also increases. Furthermore, IPTV services, such as Video on Demand (VOD), pose a challenge for these systems in efficiently handling traffic in their networks.

Due to its low cost and high scalability, peer to-peer (P2P) live streaming has been the subject of numerous studies [1-3]. P2P is the type of streaming system used, which is typically classified into two main structures: mesh and tree. In the tree structure, data is transmitted from one peer to another without explicitly requesting subsequent chunks, resulting in short transmission delays [4-5]. However, if a peer leaves the tree structure, its descendants cannot receive data, and rebuilding the structure to enable the peer to connect with another parent can be time-consuming.

To address this problem, several mesh structures have been proposed, one of which is a streaming topology that permits peers to exchange buffer maps with one another [6-7]. When a peer requests a chunk from its neighbour, it is transmitted to all neighbours at once. This approach enables a peer to maintain its desired number of neighbours even when a neighbour leaves the network, and it enables it to receive data from other neighbours as well. Despite the advantages of the mesh structure, it employs a pull-based transmission

method, where peers have to explicitly request each chunk, leading to longer transmission delays. As a result, the mesh structure experiences playback lag between the source server and peers. Peers may experience significant playback delays, depending on when they join, even though they have requested a live broadcast. Moreover, the average playback lag in the system increases as the number of peers increases because the transmission delay increases as more data are transmitted through P2P.

In a P2P streaming system, the maximum number of peer peers that each can connect with is set according to the upload capacity of each individual peer. A P2P mesh structure uses the order in which its members join to determine the maximum number of peers they can connect with. For instance, if a peer has a shorter delay in its playback, it will be more likely to join the network early. A newly joined peer is also more likely to interact with peers that have recently joined the network. This causes the system's playback lag to increase.

Our proposed method aims to reduce the average lag in P2P streaming services by enabling users to choose their neighbour peers. This method involves connecting new peers with existing ones, while considering the release of the connections between peers. By doing so, the possibility of connecting a new peer to an intermediate one with a relatively short playback time lag increases. Additionally, as the playback range of the system becomes denser, there is more data duplication among participating peers, allowing for the storage of more data and increasing the amount of information shared between users.

Data transmission to the remaining peer is permitted only until a new peer is selected in the current setup. The primary objective of this approach is to maintain the quality of the connection between two peers even after the switch to a new peer, provided that their neighbours have adequate buffer data. Our extensive simulation has shown that the proposed technique can substantially diminish the initial delay and playback latency in P2P streaming systems.

2. Literature Survey

There are various techniques that can improve the performance of peer-to-peer (P2P) live streaming systems. Some of these include network coding, chunk scheduling, and incentive mechanisms [6-7]. Research has shown that reducing the lag between the server and the peer can help improve the performance of live programs, like stock market updates. In [8], iGridMedia used a synchronized algorithm to ensure that all its peers experience the same lag. This method was able to exploit the connection between the server's bandwidth and the playback lag. In [9], a similar technique known as Elite was suggested to reduce the lag. But its effectiveness is limited by the available bandwidth and its application is only specific to certain users. Our proposal, in contrast, utilizes existing frameworks to reduce lag without requiring additional resources such as server bandwidth.

Several hybrid push-pull structures have been proposed to enhance system performance, including mesh-pull and tree-push structures. These structures offer benefits such as decreased data transmission delay and resilience against peer churn. The mTreebone framework adopts both mesh-pull and tree-push structures to link each peer. While the tree-push structure is used for data exchange, the mesh-pull structure is employed to reconstruct the tree in case a member leaves. In [10], the authors proposed a hierarchical

push-pull framework with two topologies, namely a multi-source multicast tree and a control topology. The latter is responsible for managing membership, while the former is utilized for data exchange. Numerous other push-pull frameworks have also been proposed, which can potentially enhance performance. However, maintaining these structures can be challenging.

Peer selection schemes for P2P streaming systems have been proposed, and they can be broadly classified into three categories: random selection, locality-aware selection, and QoS-aware selection. Random selection is advantageous because it provides load balancing among peers and has robust performance. However, it is not suitable for real-time applications like P2P video streaming because it does not consider network latency and upload bandwidth.

There have been various proposed solutions to address the problem of QoS-aware neighbour selection. One of these is PRO, which selects neighbours based on their upload bandwidth by considering the upload bandwidth of all its peers [11]. The study suggests that peers with lower network latency and better upload bandwidth are more likely to become neighbours [12]. The peer divides its members into different quality classes in order to select neighbours. Additionally, a tax-based approach was proposed to increase the number of neighbours [13-14].

Several schemes that are locality-aware have been proposed to reduce the transmission delay of data. These include selecting close neighbours as their neighbours through the use of peer information. The distance between peers was calculated by using RTT as the metric for determining the locality [15]. Some methods tried to determine the P2P traffic's locality by choosing the same ISP's neighbours. Some of these methods tried to determine the P2P traffic's locality by choosing the same ISP's neighbours [16-18].

Previous studies focused on selecting the closest neighbours among individuals who had not yet reached the maximum number of connectable peers. However, these studies did not consider adjusting existing connections between peers who had already reached the maximum number of connectable neighbours. There is no evidence that changing existing connections among peers who have reached the maximum number of connectable neighbours improves their performance.

3. Connection Switching Mechanism for Neighbour Selection

3.1 Drawback of Traditional Neighbour Selection Method

The proposed neighbour selection method enables the tracker server to monitor its peers and their positions in the playback queue. Initially, media servers transmit video data to their directly connected peers. Within the mesh-pull structure, each peer transmits and receives data to and from its neighbour. Subsequently, they inspect their peers' buffermap to identify the ones that possess the necessary chunks. The media servers request their neighbours to send each chunk explicitly, and then they receive it. This approach introduces a transmission delay through each intermediate peer.

In Fig. 1, peer 10 receives data from the media server through at least three intermediate peers, causing it to have a longer playback lag than peers 2, 5, and 8 which are closer to the server. As new peers join the network, they are connected to neighbour peers who have not reached the maximum number of connectable peers. Peers who joined earlier and have shorter

playback lags are more likely to be fully connected, so new peers tend to select recently joined peers as neighbours, resulting in longer playback lags for later joining peers. When a new peer joins and the maximum number of connectable neighbours is four, it checks if each peer can be its neighbour starting from peer 1. In this case, peers 9 and 10 are selected as neighbours, which results in the new peer having the longest playback lag since these peers have the longest playback lags among all existing peers

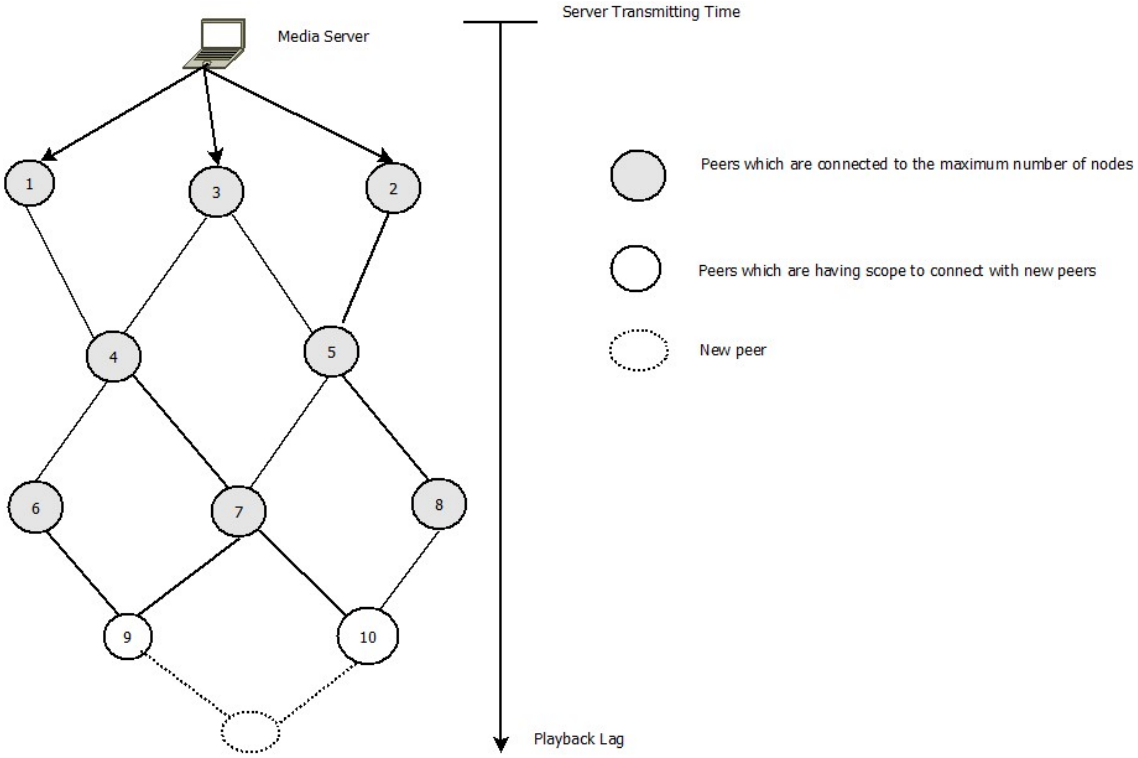

Figure 1: Existing Peer to Peer connection method when a new peer joining in the Network

Algorithm 1: Pseudo Code for Neighbour Peer selection

Begin

1. Set the first peer's index in the peer list sorted in ascending order of playback lag to current.

2. While there exists any remaining peer in the system:

a. If $N_{current}$ < max_num_neighbours:
i. Add $a_{current}$ to the neighbour peer list of Xnew (SN_{new}).
ii. Increment N_{new} by 1.
b. Else:
i. Set the first peer's index in $SN_{current}$ to neighbour of current.
ii. Set b_current_buf_enough to true.
iii. While b_current_buf_enough is true and there exists any neighbour belonging to $SN_{current}$:
1. If Bneighbor of current >= buf_threshold, set b_current_buf_enough to false and break the loop.

2. Set the next peer's index in $SN_{current}$ to neighbour of current.

iv. If b_current_buf_enough is true:

1. Set the first peer's index in $SNEG_{current}$ sorted in ascending order of playback lag to neighbour.

2. For there exists any remaining neighbour belonging to $SNEG_{current}$:

a. Set the first peer's index in $SN_{neighbour}$ to neighbour of neighbour.

b. Set b_neighbour_buf_enough to true.

c. While

b_neighbour_buf_enough is true and there exists any remaining neighbour belonging to $SN_{neighbour}$:

i. If Bneighbour of neighbour >= buf threshold, set b_neighbour_buf_enough to false and break the loop.

ii. Set the next peer's index in $SN_{neighbour}$ to neighbour of neighbour.

d. If b_neighbour_buf_enough is true:

i. Delete $a_{neighbour}$ from the neighbour peer list of $a_{current}$ ($SN_{current}$).

ii. Decrement $N_{current}$ by 1.

iii. Delete $a_{current}$ from the neighbour peer list of $a_{neighbour}$ ($SN_{neighbour}$).

iv. Decrement $N_{neighbour}$ by 1.

iv. Add $a_{current}$ and $a_{neighbour}$ to the neighbour peer list of a_{new} (Nnew).

v. Increment N_{new} by 1.

vi. Break the loop.

3. Set the next peer's index in $SNEG_{current}$ to neighbour.

c. If Nnew >= max num neighbours, break the loop.

d. Set the next peer's index in the peer list to current.

End

3.2 Proposed Connection Switching Method for Neighbour Selection

The proposed approach allows a new neighbour to join a group of existing peers, even if it exceeds the maximum number of allowed peers. This can lead to a significant increase in the number of peers, resulting in a denser distribution of their playback positions and increased data duplication. Despite the increase in the number of peers, the average playback lag remains the same as in the conventional approach.

To switch between a pair of peers in a P2P streaming system, we must first find a connection that meets the lag rate requirements. Once a suitable connection is established, both existing peers are transferred to the new neighbour. However, the new neighbours may experience playback jitter as they only receive data from the remaining peers until their buffers are sufficient. To minimize the impact on existing connections, switching is only performed after all new neighbours have sufficient buffers.

The Neighbour Selection algorithm is presented in Algorithm 1 which is for selecting a neighbour through connection switching. Upon joining the list, a new peer requests the tracker server for its list of neighbours. The tracker server maintains a list of its peers sorted in

ascending order of playback lag and periodically updates their current positions. It also sets the first peer's index in the list as current. If the number of neighbours of $a_{current}$ is less than its maximum allowed number, it will be directly selected as a neighbour of X_{new}. However, if $N_{current}$ has no neighbours in its list, the connection switching process cannot be performed effectively. In such cases, we consider two peers from each connection, i.e., $a_{current}$ and each neighbour peer of $a_{current}$ ($a_{neighbour}$), as potential neighbours of X_{new} while minimizing the degradation of playback quality.

We are checking whether the neighbours of $a_{current}$ and $a_{neighbour}$ are buffering more data than the Buf_threshold. Initially, we check if the number of buffer chunks owned by the peers in $SN_{current}$ is greater than the Buf_threshold. If it is, we then check if the buffer chunks belonging to the neighbors of $a_{current}$ in $SNEG_{current}$ (i.e., Bneighbor_of_neighbour) also exceed the Buf_threshold. It should be noted that this issue may not affect all peers whose playback lags are greater than $a_{current}$, as we have already evaluated them.

Once the aforementioned requirements are satisfied, $a_{current}$ and a_{next} can receive chunks from their respective neighbours despite their severed connection. The switching of $a_{current}$ and $a_{neighbour}$ connections to a_{new} result in a_{new} being connected to both their peers. In case N_{new} is still less than the maximum allowed number of neighbours, the sorted peer list will be searched for additional neighbours for anew. This process will continue until the maximum number of neighbours for N_{new} is attained.

Figure 2 illustrates the steps involved in selecting the neighbours of X_{new} through connection switching. When a new peer joins the network, it can have up to three neighbours. The tracker server initiates the selection process by checking if a_i, the peer with the shortest playback lag, can be a neighbour of X_{new}. As shown in the second table of Figure 2, N_i for a1 is 2, which satisfies the condition for a_i to be selected as a neighbour of X_{new}.

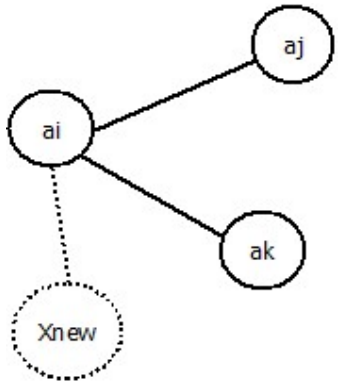

Figure 2: N_i is smaller than max_num_neigbours

In Figure 3, a_i has already reached its maximum number of neighbors, so the algorithm attempts connection switching to add a_i as a neighbor of Xnew through one of its neighbor peers. However, one of a_i's neighbour peers, (i.e. B_j), has a lower number of buffered chunks than the buf_threshold. Due to this, connection switching is not carried out as a_j may not receive enough chunks from its remaining peers (a_k and a_l) to prevent a reduction in playback quality after the connection is switched.

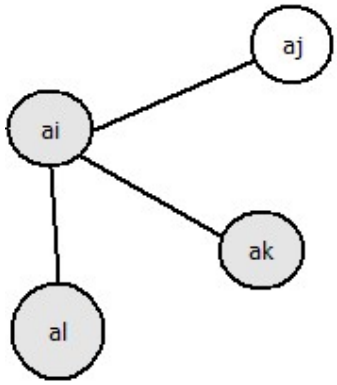

Figure 3: The buffered chunks of at least one neighbour peer of $a_{current}$ are less than buf_threshold

In Figure 4 depicts that all neighbour peers of a_i have a higher number of buffered chunks than buf_threshold. Therefore, we assess if it is feasible to switch a_i's connection with a_j to X_{new}. We conduct this assessment by verifying whether all neighbour peers of a_i and a_j are buffering more than buf_threshold. However, we observe that a_m, which is a neighbour peer of a_j, has fewer chunks buffered than buf_threshold. Thus, to prevent a decrease in playback continuity for a_j, it is decided not to conduct connection switching.

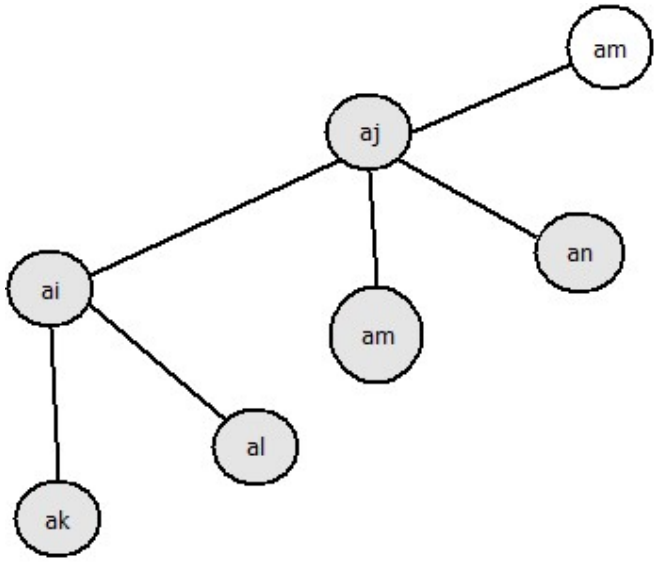

Figure 4: At least one neighbour peer of $a_{neighbour}$ has a number of buffered chunks less than buf_threshold

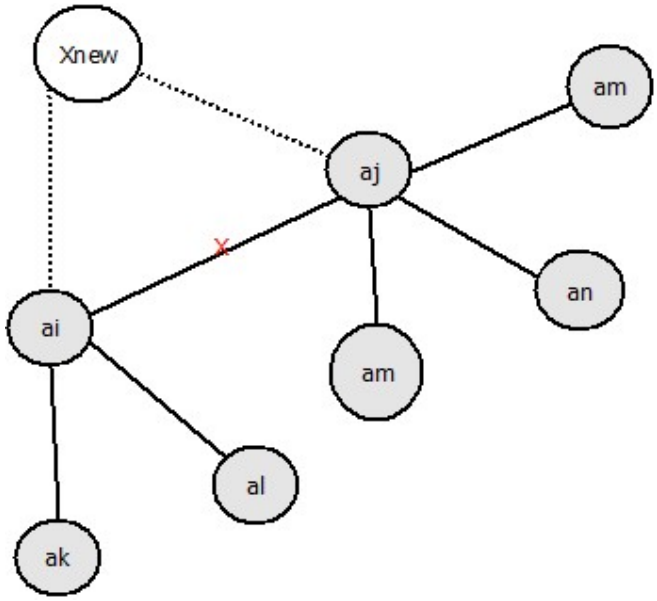

Figure 5: All neighbour peers of $a_{current}$ and $a_{neighbour}$ has a number of buffered chunks greater than buf_threshold

Figure 5 demonstrates that all neighbour peers of a_i and a_j have a number of buffered chunks greater than buf_threshold. Therefore, as all the conditions for connection switching are fulfilled, a_i and a_j establish a connection to X_{new} once their current connection is terminated.

Note that, if every existing peer has at least one pair of neighbours, Xnew must wait until one of them exits the network before initiating the switch. As soon as a neighbour departs, the new neighbour can be selected for the switch. If a new peer is not able to start its operation immediately due to the startup delay, it can still be connected to some of its neighbours. In addition, it is important to note that, even though a connection to two neighbouring peers is terminated immediately after it has joined the network, the system can still benefit from the switching process. The subsequent peer will be connected to two of its neighbours with relatively short playback time lags.

4. Simulation Results

We conducted a series of simulations to evaluate the effectiveness of our neighbor peer selection approach using the PeerSim P2P simulator, which provides a simulated P2P network environment. The simulation involved a total of 1000 peers, with a maximum of 15 peers to which the server could directly push data. The bandwidth ratios between peers and routers were set at 10-100 Mbps, resulting in 20%, 20%, 50%, and 10% for 10, 20, 50, and 100 Mbps, respectively. The backbone network was set to 8 Gbps. We assumed a playback rate of 720 Kbps for each video, with a chunk size of 30 KB and a data division rate of 3 chunks per second. The buffer map contained 512 chunks, and the tracker server updated the current playback positions of all peers every second.

To handle variations in the data receiving rate that may occur due to changes in the network condition, initial buffering is necessary. The buf_threshold value should be set to ensure that each peer can overcome these fluctuations in data receiving rate. Once a peer has buffered 45 chunks, they can begin playing the requested video, indicating that there are enough chunks to last the duration of the connection switching process. Even if a

neighbor gets temporarily disconnected, buffering 45 chunks is sufficient to prevent network jitters and ensure a smooth playback experience.

The objective of this paper is to investigate the impact of connection switching on performance. To ensure a consistent number of peers with the same number of neighbors throughout the simulation, a peer churn rate has been set. Our simulations have demonstrated that a Poisson distribution with an inter-arrival and departure time of 600 s per peer can achieve this goal.

We conducted performance testing to compare our proposed peer selection policy with the PPLive[19]. The former demonstrated better startup latency and playback continuity than the latter. PPLive is currently the most widely used peer-to-peer streaming system, but it only accepts new neighbours if their current neighbours are not at the maximum limit. To analyze the impact on performance, we varied the max_num_neighbours parameter from 4 to 9. As previously mentioned, in our approach, connection switching occurs only when all neighbours maintain a buffer greater than the buf_threshold. We weakened this condition in our simulations to study its impact on system performance.

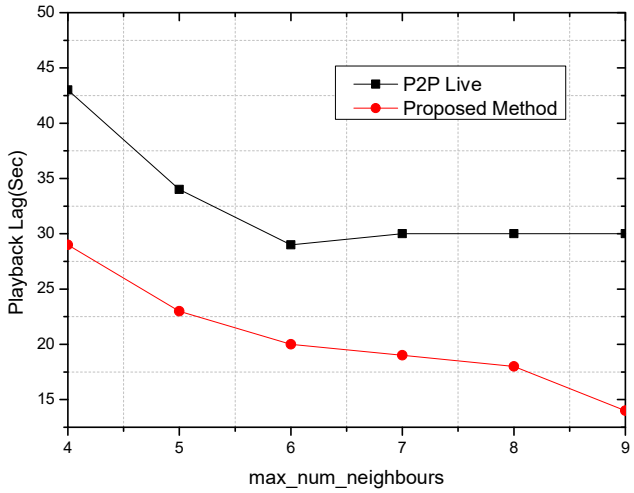

Figure 6: Playback lag Vs Max_num_neighbours

In Figure 6, the proposed scheme for selecting neighbours exceeds the standard method in terms of performance when it comes to playback lag. The proposed scheme's average playback lag is 21.8 s, while that of the conventional scheme is 28.4s. The improvement in performance is further boosted by the setting of the maximum number of neighbours to 4. The existing approach prioritizes the closest neighbours to the server's data transmission time when choosing new neighbours. This method makes it more likely for older and newer peers to be fully connected. The existing method increases the average playback lag by about a factor of one for every participating peer. On the other hand, our proposed scheme provides a choice between fully connected and unconnected peers depending on the connection condition. The selection of new neighbours causes the system's playback lag to increase significantly if the number of them is increased. This means that the system's performance is affected by the increasing number of neighbours.

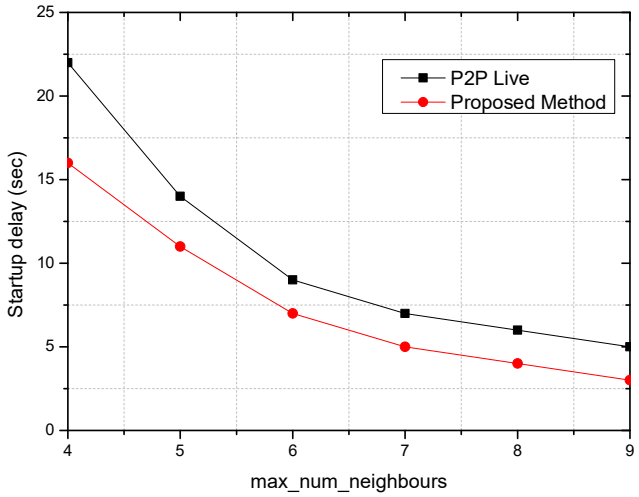

Figure 7: Startup Delay Vs Max_num_neighbours

Figure 7 shows that our proposed method for choosing neighbours would result in shorter startup delay than the existing approach. In our scheme, the preferred method would result in a lower startup delay. This is because a new peer would be more likely to connect to an intermediate peer with shorter lags. Neighbour peers can simultaneously send and receive data, which can speed up the new peer's response time. Also, connection switching occurs if the target peer's buffer is greater than the buf_threshold. This allows the new peer to access the required video content more quickly.

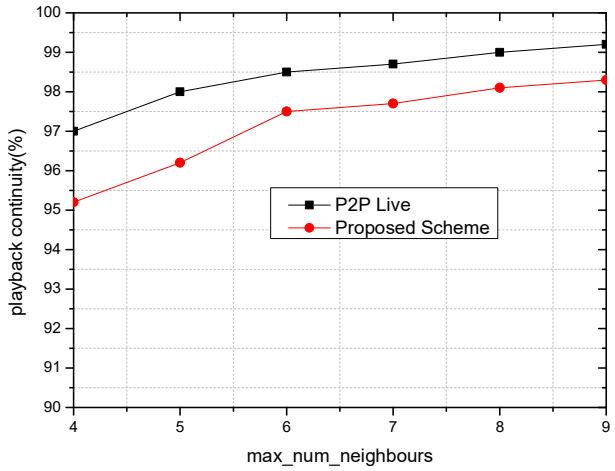

Figure 8: playback continuity Vs Max_num_neighbours

Figure 8 compares the continuity of the proposed policy with that of the conventional one when two peers are detached from their connection. The graph shows that the difference between the two schemes is small, with the average being only 0.7%. The connection switching scheme's playback continuity is 97.7%, while the conventional policy's is 98.5%. When two peers are detached during connection switching, the proposed scheme ensures that there are enough chunks to buffer the system's data. This helps

minimize the possibility of interruption. In terms of startup latency and continuity, the proposed scheme is better than the conventional one.

5. Conclusion

This paper proposed an effective method for choosing new neighbours in P2P streaming systems. This approach can help reduce the lag time between the new peer and the intermediate ones. This method leads to a significant increase in the duplication of data among the peer groups, which can cause shorter startup latency and reduced playback lag. To maintain the continuity of the system, we designed a condition that requires the two disconnected peers' neighbours to buffer the data before the connection switch occurs. Simulations show that our approach can help reduce the startup latency and minimize the lag time.

References:

[1] Zebari, Ibrahim MI, Subhi RM Zeebaree, and Hajar Maseeh Yasin. "Real time video streaming from multi-source using client-server for video distribution." In *2019 4th Scientific International Conference Najaf (SICN)*, pp. 109-114. IEEE, 2019.

[2] Deng, Jie, Gareth Tyson, Felix Cuadrado, and Steve Uhlig. "Internet scale user-generated live video streaming: The Twitch case." In *Passive and Active Measurement: 18th International Conference, PAM 2017, Sydney, NSW, Australia, March 30-31, 2017, Proceedings 18*, pp. 60-71. Springer International Publishing, 2017.

[3] Ahmad, Shakeel, Christos Bouras, Eliya Buyukkaya, Muneeb Dawood, Raouf Hamzaoui, Vaggelis Kapoulas, Andreas Papazois, and Gwendal Simon. "Peer-to-peer live video streaming with rateless codes for massively multiplayer online games." *Peer-to-Peer Networking and Applications* 11 (2018): 44-62.

[4] Zheng, Yuanhuan, Di Wu, Yihao Ke, Can Yang, Min Chen, and Guoqing Zhang. "Online cloud transcoding and distribution for crowdsourced live game video streaming." *IEEE Transactions on Circuits and Systems for Video Technology* 27, no. 8 (2016): 1777-1789.

[5] Wang, Yi-Sheng. "User experiences in live video streaming: a netnography analysis." *Internet Research* (2019).

[6] Chen, Longbin, Meikang Qiu, Wenyun Dai, and Ning Jiang. "Supporting high-quality video streaming with SDN-based CDNs." *The Journal of Supercomputing* 73 (2017): 3547-3561.

[7] Nacakli, Selin, and A. Murat Tekalp. "Controlling P2P-CDN live streaming services at SDN-enabled multi-access edge datacenters." *IEEE Transactions on Multimedia* 23 (2020): 3805-3816.

[8] ImaniMehr, Zahra, and Mehdi DehghanTakhtFooladi. "Token-based incentive mechanism for peer-to-peer video streaming networks." *The Journal of Supercomputing* 75 (2019): 6612-6631.

[9] Sammoud, Ahmed, Ashok Kumar, Magdy Bayoumi, and Tarek Elarabi. "Real-time streaming challenges in Internet of Video Things (IoVT)." In *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1-4. IEEE, 2017.

[10] Budhkar, Shilpa, and Venkatesh Tamarapalli. "Two-tier peer selection strategy to minimize delay in p2p live streaming systems." In *2016 Twenty Second National Conference on Communication (NCC)*, pp. 1-6. IEEE, 2016.

[11] Ayatollahi, Hoda, Mohammad Khansari, and Hamid R. Rabiee. "A push-pull network coding protocol for live peer-to-peer streaming." *Computer Networks* 130 (2018): 145-155.

[12] Zhang, Yuanxing, Chengliang Gao, Yangze Guo, Kaigui Bian, Xin Jin, Zhi Yang, Lingyang Song, Jiangang Cheng, Hu Tuo, and Xiaoming Li. "Proactive video push for optimizing bandwidth consumption in hybrid CDN-P2P VoD systems." In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, pp. 2555-2563. IEEE, 2018.

[13] Maheswari, B. Uma, and T. S. B. Sudarshan. "Reputation based mesh-tree-mesh cluster hybrid architecture for P2P live streaming." In *2016 3rd International Conference on Devices, Circuits and Systems (ICDCS)*, pp. 240-243. IEEE, 2016.

[14] Chang C, Chou C, Chen K, Chunk C (2014) Scheduling over swarm based P2P live streaming system: from theoretical analysis to practical design. IEEE J Emerg Sel Top Circ Syst 4(1):57–69.

[15] Yang K, Wang B, Zhang Z (2013) A method of identifying P2P live streaming based on union features. In: IEEE conference on software engineering and service science, pp 426–429.

[16] Bentaleb, Abdelhak, Praveen Kumar Yadav, Wei Tsang Ooi, and Roger Zimmermann. "DQ-DASH: A queuing theory approach to distributed adaptive video streaming." *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 16, no. 1 (2020): 1-24.

[17] Nassani, Alaeddin, Li Zhang, Huidong Bai, and Mark Billinghurst. "Showmearound: Giving virtual tours using live 360 video." In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*, pp. 1-4. 2021.

[18] Zhang, Yinjie, Yuanxing Zhang, Yi Wu, Yu Tao, Kaigui Bian, Pan Zhou, Lingyang Song, and Hu Tuo. "Improving quality of experience by adaptive video streaming with super-resolution." In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*, pp. 1957-1966. IEEE, 2020.

[19] Barekatain B, Khezrimotlagh D, Maarof M, Ghaeini H, Quintana A, Cabrera A (2015) Efficient P2P live video streaming over hybrid wmns using random network coding. Wirel Pers Commun J 80:1761–1789.