Fast Packet Recovery for PULL-Based P2P Live Streaming Systems

Houssein Wehbe, and Gérard Babonneau
Orange Labs
4, rue du Clos Courtel
Cesson Sévigné, France, 35512
Email: {houssein.wehbe, gerard.babonneau}@orange-ftgroup.com

Bernard Cousin
IRISA / Université de Rennes 1
Campus de Beaulieu
Rennes, France, 35042
Email: Bernard.Cousin@irisa.fr

Abstract— **Nowadays, Peer-to-Peer (P2P) networks play an essential role in large scale live video transmission. Though many algorithms have been proposed to deal with packet loss in P2P networks, there is still a lack of mechanisms dealing with the delay and loss constraints of live video streaming. In this paper, we propose a new loss recovery mechanism allowing the quality optimization of live video transmitted on P2P networks. Its principal feature consists in request retransmission of lost packets from a peer different of the original packet sender. This mechanism increases the probability of choosing the best available peer to make the retransmission and hence, improves the received video quality before its display time. We show by simulations that the proposed solution is efficient in comparison with the current retransmission mechanisms. This solution is independent from the sender peer selection used algorithm.**

Keywords: Peer-to-peer network; video streaming; packet loss; packet recovery; efficient; retransmission mechanism.

I. INTRODUCTION

In the last few years, multimedia data transmission over IP networks has spread enormously. High quality video transmission is becoming more and more important. However, video transmission generates constraints on the network in terms of bandwidth, latency, error rate and jitter.

The emergence of peer-to-peer (P2P) systems in the live video transmission context, also called P2P live streaming systems, enables a large performance improvement when it is compared to a centralized system, mainly in terms of scalability. The P2P principle is based on the equivalence between the roles of all the system entities called "peers". In most of these systems, the video is split into chunks, i.e., fragments, which could be either pushed by the issuer peers or pulled by the receiver ones. In PULL-based P2P systems, the video applicant initiates itself the video distribution from its owners by deciding the chunk and the peer to use. In PUSH-based P2P approach, the video owners manage the system. They decide the chunk to be sent and the destination peer. In these systems, video clients have a more passive role. In both approaches, the peers build a P2P overlay used for chunks transmission. This overlay is a P2P network built on the basic of another network, Internet network for instance. The P2P overlay construction for each peer is achieved mainly by the selection algorithms of its peer neighbors. The peers in the overlay are connected via logical links, each of which composed of a path in the underlying network. In general, peers are the end hosts in the underlying network.

To get a good video quality, a client must receive its chunks before their display times. Generally, a chunk is transmitted over the internet in several IP packets. Without forward-error-correction (FEC) technique, the loss of one single packet makes this chunk unusable. The chunk receiver cannot use it nor send it to other peers. This may degrade the video quality in a large number of peers. To guarantee the quality of service, a packet recovery technique must be applied. This technique must ensure fast packet recovery. In other words, the packets should be received before their chunk display time.

In literature, P2P live streaming systems focus on the algorithm design of overlay construction [1] or chunk exchange policy between peers [2], i.e., the choice of chunks to send (respectively receive) and its destination receiver (resp. sender) in PUSH-based (resp. PULL-based) P2P system. However, they do not propose a specific recovery technique of lost packets. Moreover, usual recovery techniques do not take into account the video temporal constraints since they have been proposed initially for file transmission. In practice, these techniques propose to retransmit the lost packets from their original sender. However, if this peer is not available during the retransmission request, the probability for not receiving retransmitted packets in time will remains very high. Which can affect the video quality.

In this paper, we propose to request retransmission of lost packets from a randomly selected peer different from the original sender. Indeed, we assume that the original sender in not always the most appropriate to achieve the retransmission. The mechanism proposed in this work aims to choose an available sender for retransmission. This choice increases the probability for receiving the lost packets before their display time and enhances then the received video quality. Our retransmission mechanism is mainly applied in PULL-based P2P live streaming systems [3]. Indeed, these systems, it is the client that manages the video streaming and then the video quality improvement. It has information on the chunks it requires and on the system peers. In loss case, it may then easily apply our retransmission mechanism. The advantage of

 ISBN: 978-1-61208-102-1

this mechanism is that it does not need adding new signaling messages, or the modification of the overlay construction used algorithms. The performance carried on the proposed mechanism shows its efficiency in comparison with current retransmission mechanisms.

This paper is organized as follows. Section 2 presents the state of the art of loss recovery mechanisms used in current P2P streaming systems. Section 3 presents our proposed solution. Section 4 introduces its performance evaluations. Finally, Section 5 concludes our works.

II. STATE OF THE ART

The loss recovery schemes used in P2P streaming systems are based on packet retransmission or FEC techniques.

Packet retransmission is a very simple and well known method in the state of the art. In transport control protocol (TCP) the video receiver must send to the sender an acknowledgement for each received packet. The sender uses the lack of acknowledgement (or several acknowledgements with the same acknowledgement number) to detect lost packets and retransmit them. But given that TCP considers the packets loss as network congestion signs, it reduces its transmission rate systematically, at loss detection. This may additionally degrade the video quality. Moreover, in asymmetric networks (like P2P networks using, for instance, ADSL links as access network) when the receiver upload channel is blocked, the acknowledgements can be lost or arrive late to the sender. In this case as well, TCP sender needlessly reduces its transmission rate. For these reasons, TCP is still not suitable for real-time video transmission.

User Datagram Protocol (UDP) does not have these drawbacks because it maintains its transmission rate, but it does not have a lost packet recovery mechanism. To apply retransmission with UDP protocol, it is necessary to define a technique dealing with packet loss detection at receiver. That allows it to request their retransmission. We can, for instance, use the Real-Time Transport Protocol (RTP) [4]. This protocol allows the sender to number the packets before sending them. The receiver can thus send a retransmission request when a packet loss is detected. However, with this simple retransmission mechanism, video receivers do not have guarantee on packet recovery time and thus on the video quality.

To solve the retransmission issue in UDP, the FEC technique is proposed. In this technique, a redundancy data is added to the transmitted stream in such a way the lost packets can be recovered by the receiver without retransmission. The live video case, video source must know the rate of loss and its pattern for adding enough redundancy data to protect the stream. However, in P2P networks, where the links are heterogeneous and nodes behavior is unpredictable, video source cannot know the loss rate throughout the whole paths. Thus, the use of FEC in these networks does not allow correcting all loss types. Besides, generally the redundancy data quantity introduced into the stream is very high, because it is permanently configured to repair the worst loss case. This may increase network congestions and therefore contribute to the packets loss.

In reality, most of the current P2P streaming systems use TCP protocol for data transport [1][6][7]. Systems using UDP protocol (for instance [8]) do not propose a specific loss recovery mechanism. An enhancement for video quality transmitted over P2P network was proposed in [9]. It consists in protecting, via FEC and/or retransmission, the most important video packets. The authors of [9] suppose that these packets may contain an image of type I and P of a group of pictures (GOP) of a MPEG video stream. Indeed, images of these two types are more important than the images of the third type (B image) because the decoding of the third type is dependent on both first ones. Also P images depend on I images. Results presented in [9] show that this protection technique improves the video quality. However, performance analyses have been carried out on a P2P system where the receiver can receive the video packets from only one peer. However, these results are only valid in this particular case. Other works have been concentrated on retransmission of pre-recorded video. They do not treat the retransmission of live video where the video is more sensitive to transmission delay [5]. The authors of [11] propose a model-based packet scheduler for P2P streaming systems with retransmission. Their proposal consists in requesting retransmission of lost packets from their original sender. This sender is chosen initially by a specific technique. Its principle is that the video receiver watches the channel state of all the peers which it knows, and then it chooses for each video packet the peer minimizing the transmission delay. That may, according to the authors, accelerate recovery in the loss case and then improve the video quality. However, this technique is not optimal and it can not be applied in the live video case. Indeed, it requires selecting a sender for each IP packet, which may in P2P live streaming systems, according to [10], generate an important loss rate and a waste of network resources. Generally, it is preferable to apply the sender selecting technique for each chunk composes of several packets. This proposal will be detail in the next sections using live streaming video.

III. THE PROPOSED RETRANSMISSION MECHANISM

In PULL-based P2P live streaming systems, video receiver decides which chunk should be requested and its sender peer. It exchanges, periodically with its neighbor signaling messages to know their available chunks, and then it chooses the chunk to request. The presence of several peers having the same video chunk raises an issue about the selection of the best peer to ensure a good quality of service for the chunk transmission. Several metrics exist which are used to select this peer, such as the available bandwidth between sender and receiver, available bandwidth of the sender node, the transmission delay between sender and receiver, i.e., one way delay, the round-trip time (RTT), the number of hops, etc. Signaling messages exchanged between peers or monitoring mechanisms allow estimation of these metrics.

Generally, a chunk is bigger than a IP packet [10]. A chunk will be sent in several IP packets. In network disturbance cases, one or some consecutive IP packets are lost. Video receiver considers a chunk unusable if at least one packet is lost. It can not display this chunk neither sends it to other peers. To improve video quality received by all the system peers, we

 ISBN: 978-1-61208-102-1

carry out a lost packet retransmission. In fact, the retransmission of some packets allows decreasing network congestion by comparing it with the chunk complete retransmission or redundancy FEC that must be sent permanently even if there is no loss in the network.

In current P2P streaming systems, the receiver requests the retransmission of lost packets from their original sender. This is the case of TCP and most of retransmission techniques proposed with UDP [9][11]. This is the peer that sends their chunks. The receiver chooses the best peer available in the network according to the selection algorithm used. This peer is thus the best which may give the chunk at time. However, after loss detection, it is not necessarily the best sender of retransmission packets. Indeed, firstly, the processing and sending capacity of this sender is, on one hand, reduced by sending others packets belonging to the chunk affected by loss. On the other hand, the capacity can be reduced by new chunk requests received by this peer since it was selected. Secondly, processing and sending capacities of other peers may have significantly increased. Thirdly, loss rate and delay characteristics can vary according to the data volume transmitted. For example, small volumes (only one packet), often undergo shorter delays than large volume. For these reasons we propose to request retransmission of lost packets from the most available peer. It will not necessarily be the original sender. In this way, the probability of receiving the retransmitted packets before their chunk display time is increasing. This is very important in live video streaming because the receiver watches live video and hence is very sensitive to the display time. Figure 1 and Figure 2 show such scenario in which the mechanism proposes in figure 2 is more efficient than current mechanism. In the scheme represented in these figures, the peer P2 needs the chunk C1 existing in peers P1 and P3. Without loss of generality, let us assume that Round Trip Time (RTT) between P2 and P1 is less than RTT of (P2-P3), that the upload bandwidth of P3 is greater than that of P1 and the chunk C1 is composed of packets (p1, p2, p3, p4, p5). Though, P1 is closer to P2, its lower upload bandwidth does not guarantee the complete reception of C1 before its display time, noted TV in Figure 1 and 2. Thus, P2 sends its request to P3. Now, let us assume that the packet p3 is lost during its transmission between P3 and P2. If P2 requests retransmission of p3 from P3, C1 will be received completely at instant T1. However, if T1 is greater than TV display time, C1 will be considered unusable. However, if P2 sends its retransmission request to P1 (Figure 2), the probability of receiving the retransmitted packet at instant T0 smaller than TV, is high. Indeed, issuing a single packet does not request a large bandwidth. Thus, our retransmission mechanism can allow the complete reception of the chunk before its display time, which improve the quality of the video display.

In practice, several metrics may be used to select the best retransmission sender RTT, peers bandwidth, etc. The estimation of these metrics requires the exchange of signaling messages periodically between peers. These messages may increase network congestion, and then may contribute to the packet loss. To limit the control messages overhead, we propose to select the retransmission sender randomly among peers having the chunk affected by loss. This retransmission

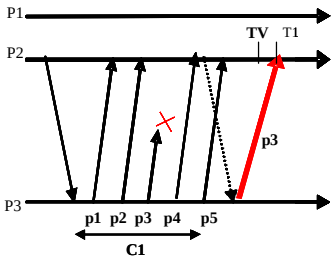

Figure 1: Video chunk transmission on P2P networks. Retransmission of lost packets is requested to the original sender.

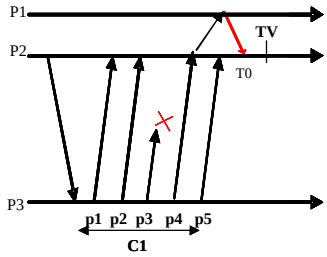

Figure 2: Retransmission of lost packets is requested to peer different from the original sender.

mechanism presents advantages in terms of flexibility and robustness. It doesn't imply any constraint on the data coding neither on network architectures. It does not need new control messages. The only condition is that the receiver has a list of peers having the chunk affected by the loss. This is always true in PULL-based P2P live streaming systems, because the receiver has this list before requesting initially the chunk. In consequence, this mechanism works properly with any PULL-based P2P system. To verify our mechanism effectiveness, we carried out a set of simulations whose results are presented in the next section.

IV. PERFORMANCE EVALUATION

Our mechanism is considered efficient if it can improve video quality in comparison with mechanisms where the receiver sends its request retransmission of the lost packets to their original sender, called hereafter "classic mechanism". A fairly high error rate will be applied. It puts the mechanisms under conditions that do not allow them to correct all the losses. Indeed, if we test the two mechanisms in situations where a correction rate of 100% is assured, it would not be possible to distinguish them.

To show our mechanism effectiveness, we carried out, using OPNET Modeler, a simulator modeling PULL-based P2P live streaming service. We explain in this following the chosen transmission algorithms.

To implement classic retransmission mechanism, we have made the following choices: (1) using peers with highest upload bandwidth, (2) request the chunks from the closest peer in term of RTT. The first choice makes, in loss case, the original sender of the lost packets more available. It has a high bandwidth, and can therefore answer to many requests (chunk transmissions or packet retransmissions) without having congestion problems. In the second choice the use of RTT aims to reduce as much as possible the packet transmission time. Applying these two choices, the probability of receiving the retransmitted packets before their chunk display time, will be increased. This improves the video quality.

Then, we need to define a chunk location technique. Its aim is to give to each peer in the system, a set of other peers and the chunks they have. In classic P2P system, this set is used to select a sender for a given chunk. This set is also used by our mechanism to randomly select a retransmission sender. But, our mechanism does not impose any constraint on the

 ISBN: 978-1-61208-102-1

technique used to build this set of peers and their chunks. Moreover in any P2P system it always exists. Any chunk locating technique may be used. We propose to use the following chunk location technique. When a new peer connects to the system, it will receive a list of peers watching the video. These peers are called "Neighbor" and the number of neighbor is identified by the parameter "Neighbor_number". Periodically, a peer exchanges with its neighbor peer signaling messages to get their available chunks. The exchange period is noted "Exchange_period". This periodic exchange exists in many P2P systems such as [6][7]. The "Neighbor_number" must be limited to reduce signaling message overhead and thus the network congestion. In our simulator, we assign random neighbors to a new peer among all the peers watching the video. They will not thus necessarily be next to the first peer in term of geographic distance or RTT. This choice ensures a load balance in the network.

"Neighbor_number" and "Exchange_period" are two important simulation parameters. They may affect the video quality even if there is no loss in the network. If "Neighbor_number" is small, the receiver will not have much choice to select the best sender for a given chunk. The probability of choosing an unavailable peer will thus be high, with the risk to hinder the chunk reception and affect the video quality. Similarly, if "Neighbor_number" is large, the number of signaling messages exchanged between peers every "Exchange_period" may generate traffic which may increase network congestion and hence packets loss. Optimal values of these two parameters are thus needed to ensure the video quality. We carried out a series of simulations to find these values and verify the validity of our algorithms before testing our retransmission mechanism.

Our simulator consists of 500 peers and a central server generating a live video. Without lack of generality, the simulations were performed with a single video since the videos are independent. The same assumptions are considered in literatures [9][12]. We used a video of 300 kbit/s, as in [12]. Peers are homogeneous and have no constraint on their download bandwidth (it is often the case in P2P system, such as [13]). To respect the choices we discussed above, we attribute to peers a large upload bandwidth. We chose a value of 2 Mbit/s. It is very large compared to the video rate. It allows each peer to serve many requests simultaneously and ensure proper dissemination of content among peers. This value is possible on FTTH network, but also on xDSL network.

A.Performance evaluation of the model without packet loss

Performance evaluation without loss of packet allows us to find the good values for "Neighbor_number" and "Exchange_period". These values will be used later to test our retransmission mechanism with packet loss.

The most important metrics to measure are:

- Chunk loss rate: A chunk is considered lost if the receiver does not receive its all packets before its display time. To measure this metric, during the simulation, each peer computes the number of video chunks to be received and the number of chunks considered as lost. Using the values computed by all the system peers, we compute at the end of the simulation, the lost chunks percentage for all video chunks during the simulation. This percentage allows us to know the system loss rate. If it is at 0%, it means that all peers have received a perfect video.

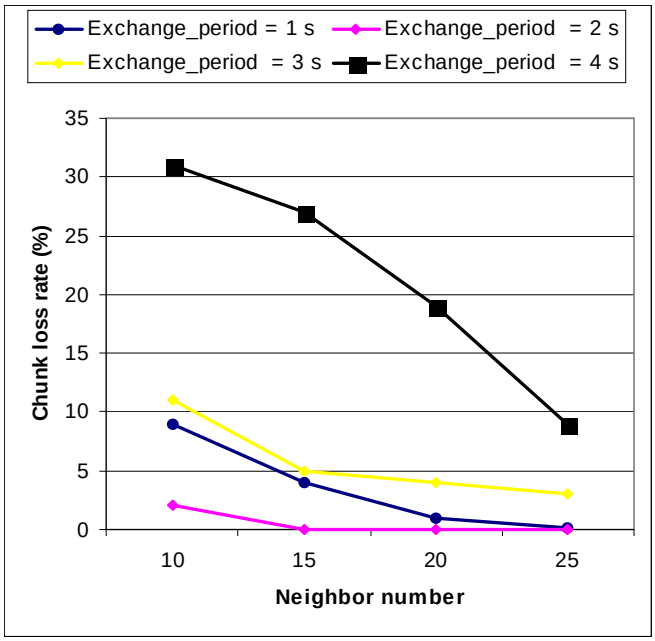

Figure 3: Chunk loss rate evaluation relative to neighbor number and signaling messages exchange period.

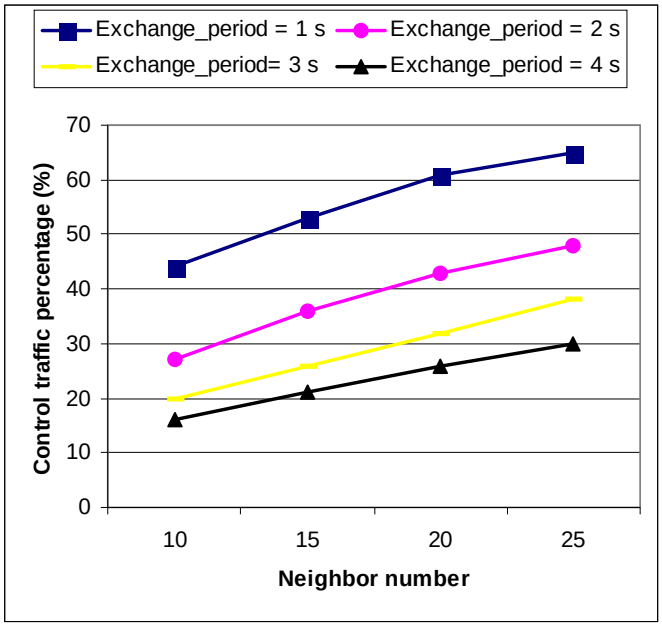

Figure 4: Control traffic percentage in all the system relative to neighbor number and signaling message exchange period.

- Control traffic percentage: This is the percentage of bytes of the signaling messages in relation to the total number of data bytes sent by the system peers (signaling + data). This metric allows us to see if the control traffic wastes the peer upload bandwidth.

 ISBN: 978-1-61208-102-1

Figure 3 shows the chunk loss rate relative to "Exchange_period" and "Neighbor_number". Since that there is no loss on the network, the reason of chunks loss is due their late reception. Figure 3 shows that the chunk loss rate decreases with the number of Neighbor increasing. Indeed, if the latter increases, the receiver has more choices to select a best sender for a given chunk. This increases the probability of finding an available sender, and reduces the probability of receiving the chunk out of delay.

However, Figure 3 shows that chunk loss rate also varies according to the "Exchange_period". When the period is larger or equal to 2 s, we remark that chunk loss rate increases with the increase of exchange period. The reason is that for a long exchange period, peers must wait some time before locating the new chunks in the system. This can delay the chunk requests and consequently the chunk reception time. The probability of receiving chunks with a delay will be large. This explains the existence of chunk loss with a large exchange period. In the case where exchange period is equal to 1 s, we can remark that the loss rate is not always 0%. For instance, this is different from the case where period is 2 s. With this short period, chunks requests will not be delayed. The reason of the loss in this case is, therefore, the non-availability of sender peers. Figure 4 gives us a verification of this fact. It assesses the control traffic percentage relative to "Neighbor_number" and "Exchange_period". Remember that each peer must exchange signaling messages with all its Neighbor at each "Exchange_period". Thus the control traffic quantity increases, thus, with the increase of "Neighbor_number". It also increases if the exchange period is decreased. Figure 4 shows this percentage variation. If the exchange period is 1 s, control traffic percentage is high; it is grater then 45%. In this case, peers send so much traffic control, which makes them less available to send the chunks. These chunks may be received out of delay. This explains the presence of loss when the period is 1 s.

It may be noted that "Neighbor_number" best value is 15 peers, when the "Exchange_period" is 2 s. These two values allow us to get the best video quality since the chunks loss rate is 0%. Thus minimizing the control traffic quantity exchanged between peers. By conducting simulations with these values, we can ensure that there is no chunk loss due to the algorithms selected such as the chunk location algorithm.

B.Performance evaluation with packet loss

To show the effectiveness of our mechanism, it is compared with classic retransmission mechanism.

We have shown in the previous section that our simulation model can guarantee a perfect video quality, if there is no loss on the network. To compare the two retransmission mechanisms, we have introduced on the links a uniform loss which rate is equal to 10% of transmitted packets. This is a very high rate compared to real network ones, but we have selected this value to show our mechanism effectiveness. Without lack of generality, we assume that signaling messages are not affected by loss, which could correspond to transport them by TCP.

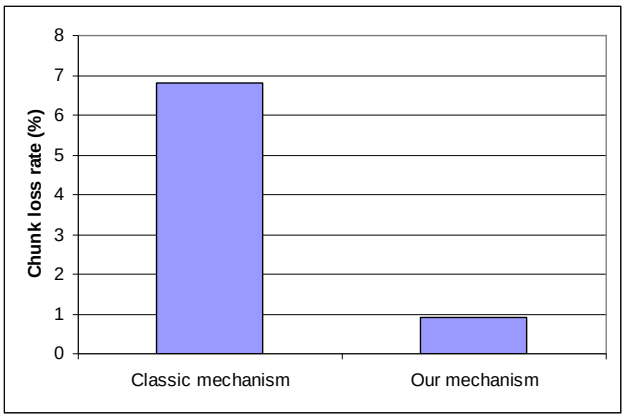

Figure 5: Comparison of the retransmission mechanism efficiency. Chunk sender is chosen relative to RTT.

Using parameter values deduced from previous paragraph, we carried out two simulations. In the first, we applied classic retransmission mechanism and in the second we applied our mechanism. A retransmission mechanism is considered efficient, if it ensures retransmitted packet reception before their chunk display time. In other words, the mechanism is considered efficient if it minimizes chunk loss rate. Remember that a chunk is considered lost if one of its packets is affected by a loss or if one of its packets arrives out of delay in relation to the display time.

Using the RTT selection parameter, we first measured the chunks loss rate with the two retransmission mechanisms. The results are presented in Figure 5. This figure shows that the chunk loss rate is 6.8% using the classic retransmission mechanism where our proposed mechanism has minimized this rate to 0.9%. These results show the effectiveness of our mechanism since almost 99% of chunks are arrived on time.

According to these results, we can notice that the chunk original sender is not necessarily the most adapted peer to make the packet retransmission in loss case. Retransmission from the original sender has not avoided the late arrival of chunks. Thus this retransmission mechanism does not guarantee the video quality. The results show also that the retransmission from a peer selected randomly among the neighbors, may improve the video quality since the chunks loss rate is very low. The proposed retransmission mechanism increases the probability of selecting an available sender peer to make retransmission. This increases the probability of receiving on time the retransmitted packets and hence improves the video quality.

In Figure 5, the transmission algorithms and simulation parameters were chosen to model the case where classic retransmission can work as well as possible. Thus we can assume that our mechanism will also be effective in any other case. To verify this assumption, we carried out simulations comparing the two retransmission mechanisms in another case. We kept the same simulation parameters and we changed the selection algorithm of sender peer. Indeed, if this algorithm ensures the choice of the best peer for a given chunk, then the retransmission from this peer can increase the probability of receiving the retransmitted packets before their chunk display time. Our mechanism shows its efficiency independently of the

 ISBN: 978-1-61208-102-1

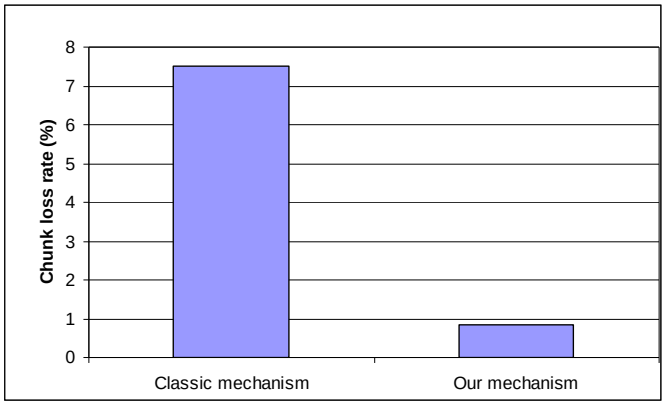

Figure 6: Comparison of the retransmission mechanism efficiency. Chunk sender is chosen randomly among the neighbors.

used algorithms. In the previous simulations, we have used an algorithm selecting the sender for a given chunk according to RTT. In the simulations presented in Figure 6, we used an algorithm based on randomly selection of this peer among peers having the chunk. We can remark that applying our retransmission mechanism, the chunk loss rate is always reduced in comparison with the classic retransmission. Thus, we assume that there is no impact of the sender peer selection algorithm on our retransmission mechanism effectiveness.

V. CONCLUSION

Today, video distribution towards a large number of receivers is a fundamental need. Appearance of P2P systems has allowed this need to be answered. Current P2P live streaming systems have low video quality. The main reason is the packet loss. This loss is due to the heterogeneous and dynamic characteristics of peers involved in the system, as well as the lack of performance guarantees in IP network.

Current P2P live streaming systems do not offer a specific mechanism to solve packet loss problem. Usual mechanisms do not take into account the packet recovery time since they are proposed initially for file transmission. In this paper, we have proposed a packet retransmission mechanism for PULL-based P2P live streaming systems. It consists in requesting lost packets retransmission from a peer randomly selected among the peers having the chunk, in general, a different peer of the original sender. We have shown that this increases the probability of receiving retransmitted packets before their chunk display time. With our mechanism, we have shown that the chunk loss rate is reduced to 0.9% improving then the video quality. The advantage of this retransmission mechanism is that it does not impose constraints nor on P2P architectures, neither on data coding. Moreover, this mechanism is independent from the sender peer selection used algorithm.

REFERENCES

[1] R. Lobb, C da Silva, A. Leonardi, E. Mellia, and M. Meo, "Adaptive overlay topology for mesh-based P2P-TV systems", 18th international Workshop on Network and Operating Systems Support For Digital Audio and Video, June 2009, pp. 31-36.

[2] P. Hoong and H. Matsuo, "Push-pull incentive-based P2P live media streaming system", Wseas Transactions on Communications, Feb. 2008, pp. 33-42.

[3] N. Magharei and R. Rejaie, "PRIME: peer-to-peer receiver-driven mesh-based streaming", IEEE/ACM Transactions on Networking, Aug. 2009, pp. 1052-1065.

[4] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RFC 3550 - RTP: A Transport Protocol for Real-Time Applications", IETF standard, July 2003.

[5] F. Pianese., J. Keller, and E. Biersack, "PULSE, a flexible P2P live streaming system", 9th IEEE Global Internet Symposium, 2006, pp. 1-6.

[6] L. Vu, I. Gupta, J. Liang, and K. Nahrstedt, "Mapping the PPLive network: Studying the impacts of media streaming on P2P overlays", Department of Computer Science, University of Illinois at Urbana-Champaign, Tech. Rep. UIUCDCS-R-2006-275, 2006.

[7] T. Do, K. Hua, and M. Tantaoui, "P2VoD: providing fault tolerant video-on-demand streaming in peer-to-peer environment", IEEE International Conference on Communications, 2004, pp. 1467-1472.

[8] M. Hefeeda, A. Habib, B. Botev, D. Xu, and B.Bhargava, "PROMISE: peer-to-peer media streaming using CollectCast", Eleventh ACM international Conference on Multimedia, 2003, pp. 45-54.

[9] B. Akabri.; H. Rabiee, and M. Ghanbari, "Packet Loss Recovery Schemes for Peer-to-Peer Video Streaming", Third International Conference on Networking and Services (ICNS), IEEE Computer Society, 2007, pp. 94.

[10] N. Hegde, F. Mathieu, and D. Perino, "Size Does Matter in Epidemic Live Streaming", Technical report 7032, INRIA, 2009.

[11] Y. Jung and Y. Choe, “Channel-adaptive packet scheduler for retransmission-based peer-to-peer stored-video streaming”, Tenth IEEE International Symposium on Multimedia, 2008, pp. 390-395.

[12] M. Zhang, Q. Zhang, L. Sun, and S. Yang, "Understanding the Power of Pull-Based Streaming Protocol: Can We Do Better?", IEEE Journal on Selected Areas in Communications, 2007, pp. 1678-1694.

[13] F. Picconi and L. Massoulie, "Is there a future for mesh-based live video streaming?", Eighth International Conference on Peer-to-Peer Computing, 2008, pp. 289-298.

[14] Z. Xinyan, L Jiangchuan, L. Bo, T. Shing, and Y. Peter, "CoolStreaming/DONet: A Data-driven Overlay Network for Peer-to-Peer Live Media Streaming", In IEEE Infocom, 2005, pp.13-17.

[15] M. Castro, P. Druschel, A. M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "Splitstream: High-bandwidth multicast in cooperative environments", Nineteenth ACM symposium on Operating systems principles, 2003, pp. 298-313.

[16] C. Zhang, H. Jin, D. Deng, S. Yan, Q. Yuan, and Z. Yin, "Anysee: Multicast-based Peer-to-Peer Media Streaming Service System", Asia-Pacific Conference on Communications, 2005, pp. 274-278.

[17] H. Luo, D. Wu, S. Ci, A. Argyriou, and H. Wang, "Quality-Driven TCP Friendly Rate Control for Real-Time Video Streaming", Global Telecommunications Conference, 2008, pp. 1-5.

 ISBN: 978-1-61208-102-1