Osama Abboud, Konstantin Pussep, Markus Müller, Aleksandra Kovacevic, Ralf Steinmetz:
Advanced Prefetching and Upload Strategies for P2P Video-on-Demand.
In the proceedings of the ACM Workshop on Advanced video streaming
techniques for peer-to-peer networks and social networking, October 2010

Advanced Prefetching and Upload Strategies for P2P Video-on-Demand

Osama Abboud, Konstantin Pussep, Markus Mueller,
Aleksandra Kovacevic, and Ralf Steinmetz

Multimedia Communications Lab
Technische Universität Darmstadt, Germany
{abboud, pussep, mueller, sandra, steinmetz}@kom.tu-darmstadt.de

ABSTRACT

The peer-to-peer (P2P) paradigm has recently shown promise in enhancing performance and decreasing costs of classical client/server Video-on-Demand (VoD) systems. Since current P2P VoD designs began with BitTorrent concepts and its school of thought, little is known about the limits of such systems in a collaborative environment. In this paper we investigate how local knowledge about neighbors' playback positions and advanced upload strategies can be used to reach these limits. Our design can be used not only to increase average user experience in terms of playback continuity but also to have more resilience against churn and flash crowds even with modest server resources. Extensive simulations support the feasibility of our approach, which, with low overhead, allows for achieving high performance and low server utilization.

Categories and Subject Descriptors

C.2 [**Computer-Communication Networks**]: Distributed Systems

General Terms

Algorithms, Design, Performance

Keywords

Peer-to-Peer/Overlay Networks, Video-on-Demand, Content Distribution

1. INTRODUCTION

Popularity of Internet Video-on-Demand (VoD) increases significantly the revenues of content providers [8]. However, the client-server technology behind current VoD systems introduces significant costs due to the increased user demand which, at a certain point, can no longer be covered by revenues when switching to High Definition (HD) content [6]. P2P VoD has become a good alternative that provides more scalable and resilient delivery at significantly decreased costs for the provider. This is achieved by shifting the load of a server to the users interested in the same video. The benefits of this approach are demonstrated by the wide usage of the P2P streaming systems Octoshape[1] and iPlayer[2] in broadcasting major events such as the Olympics or popular TV content.

While the costs are pushed away from hosting servers in P2P VoD, it must provide comparably short start-up delays and continuous playback. Nevertheless, a fundamental problem of P2P VoD is that just having collaborative peers that exchange data based on bartering relations, as done in [5], make good playback performance difficult to guarantee. In addition, linear streaming requirements often conflict with the high peer upload utilization required for cost effective VoD [1]. BitTorrent-inspired solutions therefore focus on balancing the mentioned requirements using either probabilistic or greedy piece selection techniques [9]. Such systems, however, still suffer from high start-up delays, especially for high bit-rate content [7].

In this paper we present a P2P VoD system that incorporates advanced prefetching and upload strategies. We go one step ahead of the *rarest-first* strategy suggested for piece prefetching in most VoD systems and present the *soon-most-needed* strategy that optimizes the piece distribution based on playback positions of other peers. Our results suggest that our strategies enable the system to support almost the theoretical limit of supported number of peers while having minimal overhead. More precisely, our contribution is that:

- We formulate a trade-off between number of supported peers in the system and achievable performance.
- We propose the use of a new prefetching strategy called soon-most-needed as an alternative to rarest-first.
- We present an advanced upload strategy that elegantly adapts to system load.

The paper is structured as follows: background and related work are described in Section 2. The streaming system overview and design with details on the used models and challenges are presented in Section 3. Our advanced prefetching and upload strategies are presented in Section 4 and 5 respectively. Finally, we show performance evaluation in Section 6 and conclude the paper in Section 7.

[1]http://www.octoshape.com
[2]http://www.bbc.co.uk/iplayer/

AVSTP2P'10, October 29, 2010, Firenze, Italy.

2. BACKGROUND AND RELATED WORK

In most research in the area of VoD streaming, performance was measured in terms of Quality-of-Service (QoS) metrics, namely throughput. However, we have found out that in such systems, playback performance can be low even when having good data rates. Therefore, we focus on assessing the playback performance rather than the throughput.

Since a video stream needs to be played back in sequence, P2P streaming systems should optimize in-order delivery. In addition, such systems should also optimize a good distribution of pieces through prefetching [3].

In most P2P video systems, pieces that are left to download are split into two sets: a high priority set and a low priority set. The high priority set constitutes a sliding window that contains pieces important in keeping the playback buffer full, which insures a continuous playback. In many approaches [1, 3, 7, 9], the high priority set is requested in sequential order while the low priority set is requested using the rarest-first approach. The rarest-first strategy works by prefetching those pieces that are least replicated in the neighborhood. This strategy has shown weak playback performance [7] due to the fact that it does not take into account whether the rare pieces will actually be needed soon enough by other peers. In this paper we introduce a prefetching strategy called *soon-most-needed*, that works by requesting pieces that will soon be needed by other peers.

An alternative to controlling of piece distribution is to group peers with similar playback positions with each other, as done in [4]. However, this design suffers from performance degradation for peers that do not find enough suitable partners. The closest to our work on studying the required balance between efficient piece distribution and linear streaming requirements are in [1, 3, 11]. In [3] the authors propose probabilistic piece selection strategies that address this problem, however they do not discuss how this balance can be actively controlled by the system. In [1], the authors suggest using network coding to increase possible collaboration between peers. In [11], Zhao *et al.* derive the optimal piece selection strategy for P2P live streaming systems depending on server load. Their design, however, is suited for small videos only and therefore not suitable for longer files as in the case of VoD.

An alternative to dynamically adapt the selection strategies is to have dynamic allocation of resources. This was done in [10], where the authors investigated how to adaptively allocate server resources so that to provide service guarantees in P2P VoD.

3. P2P VOD STREAMING SYSTEM

In this section, we present the P2P VoD system architecture. In addition, we mathematically derive the maximum number of supported peers for certain system and media resources.

3.1 System Architecture

We assume that a content provider has setup a modest cluster of servers for serving the users. Although theoretically P2P techniques can be used only when the server is overloaded, we focus on using servers only when necessary, i.e. when peers are about to stall. The peers optimize performance first by allowing neighboring peers to request for better distribution of pieces they need, and also by making a balance between uploading high and low priority pieces depending on system load. Further, we consider a system based on the idea of fallback servers that would inject the initial content and also make sure good performance at the peers is insured by attaching each peer to one fallback server. In case a new peer joins the system or when a peer is about to go into stalling, the required high priority pieces are provided by the respective fallback server.

3.2 System Model

We now present our system model that investigates the trade-off between server capacity allocation and achievable playback performance. The notations we use in the analysis are as follows:

- S: number of servers
- u_S: server upload capacity
- U: number of uploaders that have the whole file
- D: number of downloaders having an incomplete file
- u, d: upload and download capacity of a peer
- r: video bit-rate ($r \leq d$)
- f: average peer *prefetching factor*
- g: average peer *upload utilization*
- $d_r = f \cdot r$: required download speed ($r \leq d_r \leq d$)
- P_i^k: piece with index i at peer k
- P_{total}: total number of pieces

U is the number of uploaders that have the whole file, also known as seeders in BitTorrent-like systems. D is the number of downloaders that have none or part of the file. Of course a downloader also uploads pieces. Therefore the main difference between an uploader and a downloader is that the later might not always be able to fully utilize its upload capacity due to lack of pieces useful for other peers. Thus we define the upload utilization factor g as the ratio of the average upload speed of a peer to its upload capacity. For uploaders, this value is almost 1. Additionally, it is also close to 1 for downloaders in case of a system that exhibits a good piece diversity. The prefetching factor f is the ratio of the average download rate to the video bit-rate. This parameter represents how fast a peer should download the media file so that playback is sustainable.

The total offered system upload capacity is given by

$$u_{\text{total}} = D \cdot u \cdot g + U \cdot u + S \cdot u_S. \tag{1}$$

Since the total download speed is limited by the system upload capacity, i.e. $d_{total} \leq u_{total}$, and the total download speed is given by $d_{total} = D \cdot d_r = D \cdot r \cdot f$ we get:

$$D \leq D_{\max} = \frac{U \cdot u + S \cdot u_S}{f \cdot r - g \cdot u} \qquad \text{where} \quad \frac{\text{u}}{\text{r}} < \frac{\text{f}}{\text{g}}. \tag{2}$$

For a given number of uploaders, this equation calculates the upper limit ($D_{\max}$) of the total number of supported downloaders in relation to the prefetching and upload utilization factors. In addition, it gives rise to one of the tradeoffs we address in this paper: on one hand having peers make aggressive prefetching (high f) will lead to high upload utilization since it is more probable that more peers need some prefetched pieces. However this happens at the cost of using up more system resources. On the other hand, having too small f will lead to low upload utilization. In this paper we propose prefetching and upload strategies that, for minimal prefetching factor f, have a sufficiently high value of upload utilization g.

4. ADVANCED PIECE SELECTION

The proposed piece selection strategies proposed in this paper aim at striking a balance between optimizing in-order piece delivery and having better neighborhood piece distribution. The in-order aspect is covered by the high priority selection, while the neighborhood-wide one is covered by the *soon-most-needed* prefetching strategy.

4.1 High Priority Selection

Using this strategy peers select pieces that are important for current playback. This idea has been proposed for the majority of P2P VoD systems [9, 5]. Each peer maintains a high priority set that usually overlaps with the playback buffer. All pieces within this buffer are marked as high priority and retrieved in order.

In-order piece selection, as we will later show, is not enough. Since if this strategy is applied for all pieces, peers might suffer from starvation when skipping to playback positions where video pieces are not well distributed. This might happen quite often, especially in longer videos leading to playback performance drop of those peers. At the upload side, this approach works well for optimizing performance of specific connections, however, it is not always the best strategy for the whole neighborhood or the system.

4.2 Soon-Most-Needed Prefetching

In a normal P2P VoD system, different peers are not aware of the playback positions and the high priority pieces required by neighboring peers and the network. To overcome this problem, we propose to use a new piece prefetching strategy called *Soon-Most-Needed (SMN)*. This strategy optimizes the neighborhood by taking into account information inferred from the playback positions of neighboring peers by allowing the peers to actively vote for the pieces they need.

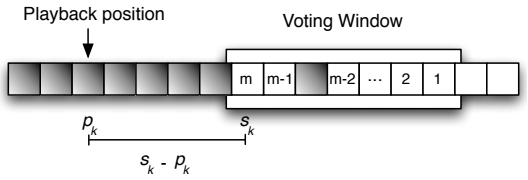


Figure 1: Voting procedure at peer k

In Figure 1, we present how voting is performed at a certain peer. Each peer is allowed to vote for m pieces that are still missing from its video file every T seconds, called the refresh interval. We suppose that voting is synchronous, i.e. all votes are received at the same time.

The vote values will be set in such a way to be decreasing with increasing piece number. In addition, voting will be for pieces starting from piece s_k, which is the first non-received piece in linear order at peer k. To allow for prioritized allocation, these values are further scaled by the difference between the playback position and the first piece being voted for. Therefore, peer k will vote for the successive m pieces not received so far according to the equation:

$$Vote(P_i^k) = (m-i)/(s_k + i - p_k) \quad i = s...s+m' \qquad (3)$$

where m' is m plus pieces already buffered within the voting window. After each peer has caste its votes, it sends the SMN list with the vote values to all peers in its neighborhood. The SMN list is sent only to downloaders and not to uploaders, since the uploaders have the whole media file and need not prefetch any pieces. The votes are then collected and aggregated by the downloader peer. Aggregation is then simply the adding up all votes for each piece, so

$$Vote(P_i) = \sum_{k=0}^{|L|} Vote(P_i^k). \qquad (4)$$

All the downloaders are continuously discovering a list of pieces that are available within their neighborhoods and also generating an SMN list. The intersection of both lists is generated. The result is a list of the SMN scores of all pieces that are available at neighboring peers. From this list, all pieces the local peer already has downloaded are removed.

The next step before prefetching can start is taking into consideration local availability of pieces. One piece could be well distributed in the local neighborhood while another one with an almost equal SMN score is very rare. In this case the rarer piece is selected. To accomplish this, the unavailability factor a is introduced. It is calculated by:

$$a(P_i) = 1 - \frac{Replicas(P_i)}{\text{size of neighborhood}}, \qquad (5)$$

which is high when there are only a few replicas for a certain piece, otherwise it is low. The SMN scores are multiplied with the availability factor and then transferred into download probabilities by normalization as follows:

$$Probability(P_i) = \frac{Vote(P_i) \cdot a(P_i)}{\sum_{j=0}^{P_{total}} Vote(P_j) \cdot a(P_j)}. \qquad (6)$$

The downloader peer then chooses its prefetching pieces according to these probabilities. If there is no peer in the neighborhood that has any prefetching piece, that is when $\sum_{j=0}^{P_{total}} Vote(P_j) \cdot a(P_j) = 0$, rarest-first is then used to build up a strong distribution of pieces. It is worth mentioning that at steady state this equation not only optimizes for playback continuity at the peers but also allows for matching piece demand to piece supply.

5. UPLOAD STRATEGIES

After each peer makes a decision on the next pieces to download, it sends a request to its neighboring uploaders. In this case, a certain uploader would continuously receive requests for different pieces and therefore has to make a decision on which piece(s) to upload. Since the requests have the distinction of being either high priority or low priority, this distinction is to be considered while making the upload decision. Based on this design, all low priority and high priority requests are put at the uploader in a high and low priority sets respectively. Now we present possible upload strategies that the uploader can use for the different sets and how they can be combined.

High Priority Set. To optimize the local connections an uploader can simply order all high priority requests according to their priority, i.e. their deadline. The closer the deadline of a certain piece, the sooner it should be uploaded. In case the deadline of a certain piece cannot be met, a quick decline message is sent to allow for fast request retry.

Low Priority Set. In addition, an uploader is able to optimize the distribution of useful pieces within its neighbor-

hood. Therefore, the uploader can favor low priority pieces according to the presented SMN prefetching strategy. This is important when the system is either in deficit state or on the verge of being overloaded, and therefore should reorder priorities to have a more robust distribution of soon-most-needed pieces so that the peers can serve high priority pieces in the near future.

Upload Bias Probability. Since system resources can be either in surplus or deficit, local and neighborhood-wide optimizations have to combined and balanced. To do this, we introduce the upload bias probability p used when making an upload decision, which will address the trade-off between optimizing performance of individual requestors and optimizing the neighborhood and therefore the system. Therefore, a peer will decide on uploading a high priority piece with probability p and on uploading a prefetching piece with probability $1 - p$.

5.1 System Load Adaptation

In a P2P VoD system, and for a given server capacity, performance is dependent on the number of downloaders, which degrades drastically near the limit $D_{\max} = \frac{U \cdot u + S \cdot u_S}{(f \cdot r - g \cdot u)}$ as presented in Equation 2. One interesting question that arises is how to minimize performance degradation during flash crowds, and also how to prepare the system for this situation.

We propose to do this as follows: the parameter p is the key to the balance between high priority and soon-most-needed pieces. Therefore, we assign the value p depending on the ratio of demand posed by the downloaders to the supply provided by them. The higher this ratio, the less capable is the server to serve high priority pieces. Thus, the uploaders and downloaders themselves should be more ready to serve such pieces. On the other hand, when the demand is much less than the supply, then the server is able to serve any required high priority request, and thus downloaders should focus on building strong distribution of soon-most-needed pieces in preparation for a flash crowd. Therefore, and based on the system model presented in 3.1, we assign

$$p = \frac{\text{demand by downloaders}}{\text{supply from downloaders}} = \frac{D \cdot r \cdot f - U \cdot u - S \cdot u_S}{D \cdot u \cdot g}. \quad (7)$$

It is worth noting that the above equation should saturate to 1 at the system limit of supported peers ($D_{\max}$), after which only high priority pieces are served by the whole network, and the system is almost in deficit. However, when $p < 1$, the system elegantly adapts to the number of downloaders while taking resources and system capacity into account. All required information to calculate a new value of p is directly available at the tracker. For simplicity, the tracker can use the average values of g and f. Finally, the tracker can send p piggy-backed within other status messages that are regularly sent to the peers without inflicting any additional overhead.

6. EVALUATION

We now assess the impact of our proposed prefetching and upload strategies. We have implemented a P2P VoD streaming system with our strategies in an extended version of OctoSim simulator [2], which is a discrete event-based simulator modeling piece transfers.

6.1 Setup and Metrics

The basic setup used for the performance evaluation is shown in Table 1. It models a VoD scenario where a content provider offers relatively long video clips, such as television series or movies (which are of major interest in a revenue-based content distribution). The content provider is interested in guaranteeing playback performance up to a certain number of users depending on server capacity and content properties. It is also interested in maximizing revenues by minimizing traffic it is serving as described in Section 3.

Table 1: Basic setup.

Parameter	Value
Simulation duration	210 minutes
Video length	60 minutes
Video bit-rate	512 kbps
Available servers	4
Server capacity (up)	4086 kbps
Peer capacity (up)	256 kbps
Peer capacity (down)	1024 kbps
Playout buffer size	7 seconds
Piece size	64 KB
Number of pieces	1800
Neigborhood size	16
Online after video playback	10 minutes

We use a peer bandwidth model with uniform values for all peers with 1024 kbps download and 256 kbps upload link capacities, which are reported to be the average speed of today's Internet users [2].

The main scenario we use to evaluate our algorithms is based on a double flash crowd situation which is often met in real VoD systems. The scenario, which is shown in Figure 2, is based on the idea that people go home at 6pm, they would start streaming some series or similar content creating some flash crowd effect. After some drop in active users, another flash crowd takes place at around 8pm which is known as the primetime, where many users join the system to watch a video content. In our simulation, 120 and 180 peers join the system during the first and second flash crowds respectively, resulting in a total of 280 peers. Both crowds follow an exponential distribution.

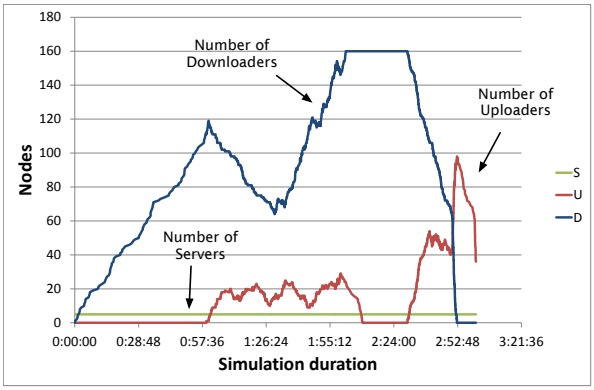
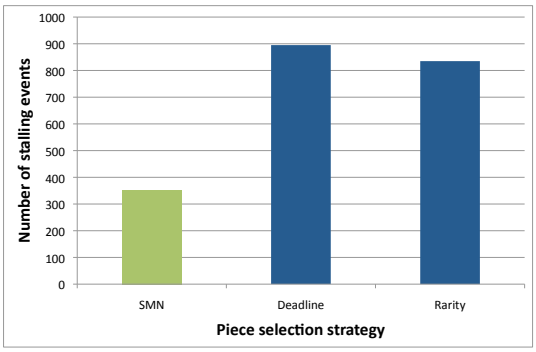
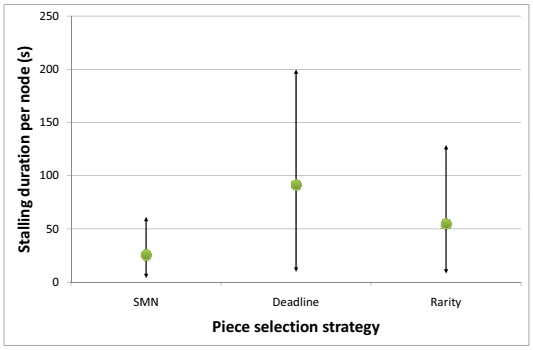


Figure 2: The double flash crowd scenario

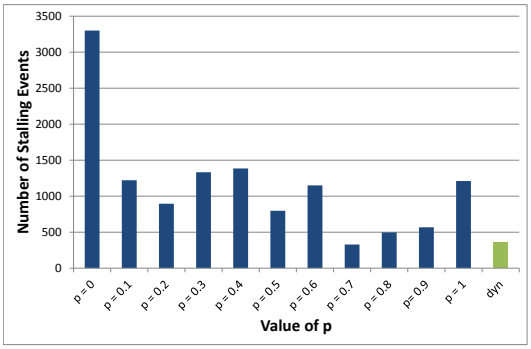


(a) Number of stalling events for all peers over the whole simulation run

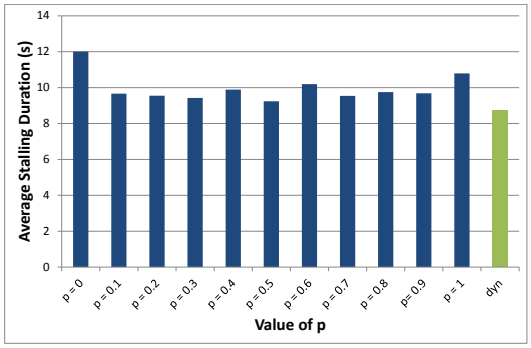


(b) Average, minimum and maximum stalling duration per stalling event

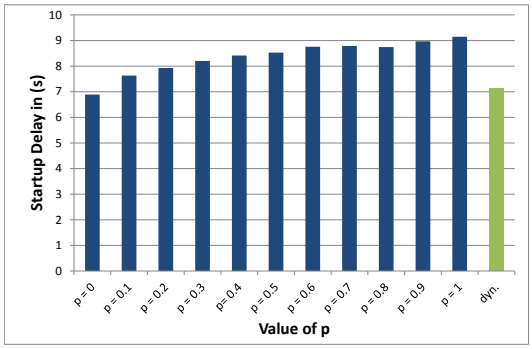
Figure 3: Performance comparison of the SMN, deadline and the rarest-first prefetching strategies



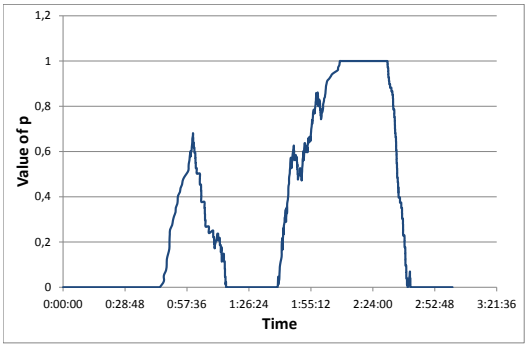
(a) Number of stalling events for all peers over the whole simulation run



(b) Average stalling duration per stalling event



(c) Average startup delay



(d) Assigned upload bias probability

Figure 4: Evaluation and impact of static and dynamic upload bias probability p on system performance

6.2 SMN versus Rarest-first

To have a fair comparison, piece selection of high priority pieces is based on an in-order selection, while perfecting can be either based on SMN or rarest first strategy. The results of this comparison are shown in Figure 3, where Figure 3a shows the number of stalling events for the whole simulation, and 3b shows average stalling duration for each stalling event. We found that using SMN resulted in better performance than rarest-first, which in turn performed better than deadline. Although the fact that deadline-based prefetching performs the worst might be surprising at first, it makes sense mainly due to the weak piece distribution this strategy inflicts. Such performance was also reported in [6].

Comparing SMN and rarity, rarity generated 835 stalling events while SMN generated only 353 (almost 50% better performance). In addition, the average duration of each stalling event is 45% shorter when using SMN. The performance advantage for SMN can be explained by the fact that this strategy prepares the system, already from the beginning, for flash crowds by prefetching those pieces that will soon be needed by the neighborhood. Therefore, peers already longer in the system have good piece distribution in

terms of how much those are needed. Hence, those peers would offload servers by exchanging those pieces already prefetched, while servers can serve new comers until they can start prefetching and support other peers.

6.3 Upload Strategy

The second set of simulations aims at evaluating the proposed upload strategies. We compare the adaptive strategy that controls the probability p (as presented in Section 5.1) with static upload bias probability p ranging from 0 to 1 with 0.1 increments. As can be seen in Figure 4, playback performance indicators (i.e. prebuffering duration, number of stalling events, and average stalling duration) show better performance when using our adaptive strategies.

In addition, Figure 4.d shows the evolution of the value p versus different simulation times. As we can see, the dynamic p allows the system to elegantly adapt to system load, and therefore make the peers more involved in supporting other peers and the system. It is worth noting that, although the adaptive upload strategy does not perform impressively better than the best static p, the adaptive p is always synchronized with the best static one with different scenarios. The relation between the best static p and specific scenario parameters has still to be further investigated.

6.4 Overhead and Performance

In this paper we have shown how the performance of a P2P VoD system can be drastically enhanced using advanced prefetching and upload strategies. In this section, we assess the overhead and performance of our algorithms.

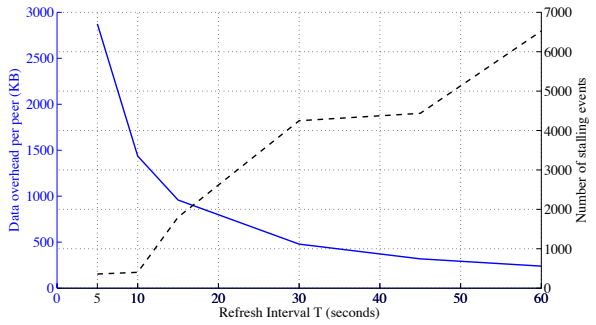


Figure 5: Overhead and performance

The results are presented in Figure 5. For different values of the refresh interval T, we show the total traffic overhead per peer in KBytes (overhead) and the total number of stalling events for all peers (performance). It is visible that the less often voting cycles are performed, the less the overhead is, and the more the performance degrades (more stalling events). The best value of T can be chosen depending on the scenario or even dynamically assigned. In any-case, the overhead for the case of $T = 5$ seconds relative to the total exchanged video data is $\frac{2900}{115200} = 2.5\%$. Additionally, increasing the refresh interval from 5 to 10 seconds does not inflict substantial performance degradation while the overhead is almost halved.

7. CONCLUSION

In this paper we have investigated mechanisms of an optimized P2P VoD based on advanced prefetching and upload strategies. We have pushed P2P VoD systems to limits by optimizing the system's playback performance while using local knowledge. Our advanced prefetching and upload strategies performs 50% better than the research standard rarest-first. The proposed optimizations can be used in existing P2P VoD systems systems that have limited server resources and flash crowd issues.

As future work, we plan to combine our strategies with prediction-based techniques to better harness the power of prefetching and have more resilience against flash crowds.

8. ACKNOWLEDGMENTS

This work was funded by the Federal Ministry of Education and Research of the Federal Republic of Germany (support code 01 BK 0806, G-Lab).

9. REFERENCES

[1] S. Annapureddy, S. Guha, C. Gkantsidis, D. Gunawardena, and P. Rodriguez. Is high-quality vod feasible using p2p swarming? In *WWW '07*, pages 903–912, New York, NY, USA, 2007. ACM.

[2] A. R. Bharambe, C. Herley, and V. N. Padmanabhan. Analyzing and improving a bittorrent networks performance mechanisms. In *25th IEEE International Conference on Computer Communications INFOCOM 2006*, pages 1–12, April 2006.

[3] N. Carlsson and D. L. Eager. Peer-assisted on-demand streaming of stored media using bittorrent-like protocols. In *In Proc. of NETWORKING'07*, pages 570–581, Berlin, Heidelberg, 2007. Springer-Verlag.

[4] B. Cheng, L. Stein, H. Jin, and Z. Zhang. Towards cinematic internet video-on-demand. In *Proceedings of Eurosys '08*, pages 109–122, New York, NY, USA, 2008. ACM.

[5] P. Garbacki, D. H. J. Epema, J. Pouwelse, and M. van Steen. Offloading servers with collaborative video on demand. In *7th Int. Workshop on Peer-to-Peer Systems (IPTPS'08)*, Tampa Bay, FL, February 2008.

[6] C. Huang, J. Li, and K. W. Ross. Can internet video-on-demand be profitable? In *SIGCOMM '07*, pages 133–144, New York, NY, USA, 2007. ACM.

[7] J. Mol, J. A. Pouwelse, M. Meulpolder, D. Epema, and H. Sips. Give-to-Get: an algorithm for P2P VoD. In *MMCN*, 2008.

[8] Parks Associates. Internet video: Direct-to-consumer services. Online: http://www.parksassociates.com/.

[9] N. Parvez, C. Williamson, A. Mahanti, and N. Carlsson. Analysis of bittorrent-like protocols for on-demand stored media streaming. In *SIGMETRICS '08*, pages 301–312, New York, NY, USA, 2008. ACM.

[10] K. Pussep, O. Abboud, F. Gerlach, R. Steinmetz, and T. Strufe. Adaptive server allocation for peer-assisted video-on-demand. In *Proceedings of the 24th IEEE International Symposium on Parallel and Distributed Processing IPDPS 2010*, Apr 2010.

[11] B. Zhao, J. Lui, and D. Chiu. Exploring the optimal chunk selection policy for data-driven p2p streaming systems. *The 9th International Conference on Peer-to-Peer Computing*, Jan 2009.