Peer-assisted On-demand Video Streaming with Selfish Peers*

Niklas Carlsson[1], Derek L. Eager[2], and Anirban Mahanti[3]

[1] University of Calgary, Calgary, Canada
[2] University of Saskatchewan, Saskatoon, Canada
[3] NICTA, Sydney, Australia
niklas.carlsson@cpsc.ucalgary.ca, eager@cs.usask.ca,
anirban.mahanti@nicta.com.au

Abstract. Systems delivering stored video content using a peer-assisted approach are able to serve large numbers of concurrent requests by utilizing upload bandwidth from their clients to assist in delivery. In systems providing download service, BitTorrent-like protocols may be used in which "tit-for-tat" policies provide incentive for clients to contribute upload bandwidth. For on-demand streaming delivery, however, in which clients begin playback well before download is complete, all prior proposed protocols rely on peers at later video play points uploading data to peers at earlier play points that do not have data to share in return. This paper considers the problem of devising peer-assisted protocols for streaming systems that, similar to download systems, provide effective "tit-for-tat" incentives for clients to contribute upload bandwidth. We propose policies that provide such incentives, while also providing short start-up delays, and delivery of (almost) all video frames by their respective playback deadlines.

Keywords: BitTorrent-like systems, peer-assisted streaming, tit-for-tat.

1 Introduction

Peer-assisted content delivery techniques are increasingly being adopted by media companies. For example, in April 2008 it was reported that the BBC iPlayer service[1], which in part uses peer-assisted delivery, was being used for downloading more than one million BBC programmes each week.[2]

When a download-and-play approach is used (as in the BBC iPlayer service), BitTorrent-like protocols [1] may be used in which a tit-for-tat policy provides incentives for clients to contribute their upload bandwidth.[3] Tit-for-tat is effec-

* This work was supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada and by the Informatics Circle of Research Excellence (iCORE) in the Province of Alberta.

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

[2] http://technology.timesonline.co.uk/tol/news/tech_and_web/article3716781.ece

[3] BBC iPlayer also offers a streaming service, but this does not use peer-assisted delivery.

tive in this context because the protocol's "rarest first" piece selection policy makes it highly likely that peers will have differing sets of file pieces and are thus able to carry out two-way mutually beneficial piece exchanges.

Clients may, however, prefer services in which they can begin viewing a video shortly after beginning download, with only a short start-up delay. In such on-demand streaming systems, once playback begins, reception of each subsequent frame must occur before the frame's play point if impairment in playback quality is to be avoided. Unfortunately, the in-order requirements of playback fundamentally conflict with the goal of high piece diversity, as needed for effective use of tit-for-tat. In fact, all prior peer-assisted protocols for on-demand video streaming have relied, for good performance including low start-up delay, on peers at later video play points uploading data to peers at earlier play points that do not have data to share in return.

This paper considers the problem of improving quality of service in peer-assisted on-demand streaming systems while retaining the basic tit-for-tat behaviour of BitTorrent-like protocols. Tit-for-tat is one of the cornerstone ideas supporting fairness and scalability in BitTorrent and has the advantage of being fully decentralized [1]. We attempt to achieve this goal through design of new, tit-for-tat compatible, peer and piece selection policies, focusing for the most part on policies to be used for server-to-peer uploads (for which tit-for-tat behavior is not an issue) rather than peer-to-peer uploads. We find that the best system performance is achieved with a peer selection (by the server) policy that preferentially allocates server bandwidth for uploads to peers at imminent risk of receiving data too late for playback, and secondly for uploads of rare pieces to newly arrived peers. Simulations of these new policies are used to evaluate their effectiveness. Our results show that substantial improvements in quality of service are feasible while ensuring that the piece diversity is sufficient for peers to effectively employ tit-for-tat.

The remainder of the paper is organized as follows. Related work is discussed in Section 2. Section 3 describes a baseline tit-for-tat based peer-assisted streaming protocol. New piece selection policies, and a server policy for prioritizing upload requests from peers, are discussed in Section 4. Section 5 describes the simulation model used for evaluating the new policies. Section 6 presents performance results. Conclusions are presented in Section 7.

2 Related Work

There exists a large literature on peer-assisted on-demand streaming systems. A significant portion of this literature has focussed on *explicit allocation* of peer upload bandwidth [2,3,4,5,6,7,8,9,10]. This literature, for example, includes tree-based cache-and-relay approaches [4,5], and work that considers the problem of determining the set of servers (or peers) that should serve each peer, and at what rate each server should operate [6,7]. In recent work, Parvez *et al.* show that systems in which pieces are retrieved in-order can significantly benefit from peer selection policies that bias uploads towards peers with fewer potential uploaders (i.e., peers requiring data closer to the end of the file) [9]. Such bias results in a natural "daisy-chain" effect wherein peers upload the most pieces to peers with slightly fewer in-order pieces than the peer itself has obtained thus far. Peer selection bias towards peers with similar playback points has also been used in an implementation of a video-on-demand system [10]. While effective in cooperative environments, we note that these techniques do not allow for effective use of tit-for-tat as younger peers typically are not able to upload to older peers.

Other work has proposed using feedback mechanisms to gain information about the upload contributions of peers at earlier playback points, and use this information to reward peers that forward pieces at a higher rate [11]. We note that such feedback-based schemes are significantly more sensitive to cheating peers than tit-for-tat techniques in which the peers themselves can effectively measure the rates they receive from other peers.

There has also been related work on piece selection policies that attempt to mediate the conflict between high piece diversity and the in-order requirements of playback (e.g., [12,13,14,15,16]). For example, Annapureddy *et al.* [12] propose splitting each file into fixed-sized segments, each consisting of some number of consecutive pieces. Segments are downloaded sequentially using a BitTorrent-like protocol. To increase the likelihood that peers downloading the same segment have pieces to exchange they propose using distributed network coding within segments, and pre-fetch some smaller number of pieces from future segments. Probabilistic piece selection policies have also been proposed [14,15,16]. Note that in order to achieve low start-up delays, these policies depend on older peers uploading to new peers that most likely do not have any needed pieces to offer in exchange.

Finally, we note that most prior work on peer-assisted video-on-demand has assumed that peers begin playback after buffering a *fixed* amount of data, or evaluate protocols with respect to the *lowest possible* start-up delay a peer could have chosen such that the (unknown *a priori*) download completion time of every piece is no later than its playback point. In contrast, we use a simple *online* rule (based on LTA [14]) to determine when playback can safely commence, and evaluate our policies with regards to both the *actual* start-up delay, as determined by this online rule, and the percentage of pieces that are not received by their playback point, given the chosen start-up delay.

3 Baseline Protocol Using Tit-for-Tat

We consider peer-assisted on-demand streaming systems with a single server and varying numbers of active peers, and focus on the delivery of a single video file. This file is divided into fixed-size pieces; each piece is further divided into subpieces as in other BitTorrent-like systems [17,18]. The unit of upload/download is the subpiece. When a peer acquires (all of) a piece, it can advertise this to other peers, and upload subpieces to other peers that do not yet have (all of) the piece. A peer may download multiple different subpieces of a piece in parallel from multiple other peers. We further assume that pieces are grouped into fixed-sized segments, as might be needed in systems utilizing coding, although the simulation results that we present here are for uncoded content delivery.

A peer is considered to be *interested* in another peer if the latter peer has at least one piece that the former peer has not yet received. A *piece selection* policy is used to select among the pieces that are available from a particular peer. In our baseline policy, the candidate pieces are ranked according to how soon they will be needed for playback, and a piece is selected by sampling from a Zipf probability distribution [14]. More specifically, with the Zipf(θ) policy, a peer j about to request a piece from peer i selects a piece k from the set of pieces that i has, but that j does not have, with a probability proportional to $1/(k + 1 - k_0)^\theta$, where k is the index of the piece, and k_0 is the index of the first piece that peer j does not yet have. While the Zipf parameter θ can be tuned so that the policy is more or less aggressive with respect to its preference for earlier pieces,[4] for the results presented here θ is fixed at 1.25. The Zipf(θ) policy has been shown to achieve a good tradeoff between high piece diversity and sequential progress. In contrast to in-order policies (including segment-based in-order versions [12]), or proposals that make explicit allocation of peer upload bandwidth so that older peers upload to newer peers, Zipf-based policies allow for effective use of tit-for-tat.

As with BitTorrent, each peer establishes persistent connections with a large set of peers but only uploads to a limited number of peers at each time instance. A *peer selection* policy determines which peers to upload to, among the interested peers. A *tit-for-tat* peer selection policy is assumed, wherein upload priority at a peer is given to those other peers that are providing the highest download rates to that peer. Periodically, a new peer is chosen to upload to, in the chance that it may offer a better download rate than the current peers. With this *optimistic unchoke* component, a random selection is made among the interested peers to which the peer is not already uploading, if any. At the server, the baseline peer selection policy is random. The above policies are commonly used in simulations of BitTorrent-like systems and provide a baseline for comparison.

4 New Policies

4.1 Acquiring Rare Pieces

For good performance including low start-up delay, all previous policies for peer-assisted streaming delivery depend on older peers uploading to new peers that most likely do not have any needed pieces to offer in exchange. Such behavior is not a concern in tit-for-tat based download systems, since in that context rarest-first piece selection can be used, and new peers can quickly acquire pieces needed by many others. It *is* a potential concern, however, in tit-for-tat based streaming systems. Even with probabilistic piece selection policies such as that used in our baseline protocol, it may take a relatively long time for new peers to acquire pieces needed by many others. Selfish peers may therefore be motivated

[4] For example, note that $\theta = 0$ and $\theta \to \infty$ yield random and in-order piece selection, respectively.

to adopt optimistic unchoke policies discriminating against new peers. Also, the upload bandwidth of new peers may be underutilized, particularly in low to moderate request rate scenarios in which there is frequently only a single or a small number of concurrently active new peers. Finally, owing to use of tit-for-tat, new peers may receive a relatively small share of the aggregate upload bandwidth of other peers, resulting in a relatively low total download rate and lengthened start-up delays. We address this concern with *rare piece delivery to new peers* (RPNP), which entails two modifications to the baseline protocol.

First, when the server unchokes a "new" peer (specifically, a peer that has not yet begun playback), rather than using the Zipf(θ) piece selection policy used in the baseline protocol, the selected piece is the rarest piece not currently being uploaded by the server to some other peer. If there are multiple rarest pieces, ties are broken randomly except when only the server has these pieces, or when the server has sufficient upload bandwidth to serve every currently active peer at the play rate, in which case ties are broken using the Zipf(θ) policy.

Second, to more quickly disseminate rare pieces to new peers, the server gives upload priority to peers that have not yet begun playback.[5] Among such peers, higher priority is given to peers for which the server has uploaded less data. The server uploads to only the n highest priority peers, where n is the number of server upload connections; ties are broken randomly.

4.2 Prioritizing Urgent Piece Downloads

We further modify the baseline protocol so as to increase the likelihood that each piece is received by its scheduled playback point, by prioritizing delivery of the next required piece for any peer that has started playback and for which this next required piece is within either the current segment being played back, or the next segment (i.e., the peer is in a "low-buffer" state).

This prioritization is accomplished through two policy modifications. First, peers in the low-buffer state use in-order piece selection, rather than Zipf-based piece selection. Second, the server gives the highest upload priority to those peers that are in the low-buffer state. The remaining peers may be prioritized as in RPNP. Among those peers in the low buffer state, higher priority is given to peers for which the server has uploaded less data. As with RPNP, the server uploads to the n highest priority peers, with ties broken randomly.

When used together with RPNP, we call this approach *urgent piece prioritization with rare piece delivery to new peers* (UP/RPNP). Note that neither RPNP nor UP/RPNP alter the tit-for-tat peer selection policy used by peers.

5 Simulation Model

We use an existing event-based simulator of BitTorrent-like systems [14]. For the results presented here it is assumed that: (i) all active peers have

[5] Both here, and in Section 4.2, we assume that the server is able to reliably identify peers to which it wishes to give preferential treatment. For example, the system may require use of content provider software with this functionality built in.

connections with each other[6], (ii) there are sufficiently many sub-pieces per piece that parallel download is always possible when multiple peers have a desired piece, (iii) a peer i (or the server) uses at most n_i concurrent upload connections, (iv) connections are not choked in the middle of an upload, and (v) new downloads are initiated only when the download bandwidth capacity D_i is not being fully utilized.

The set of peers that a peer i (or the server) is uploading to may change when (i) the peer completes the upload of a piece, or (ii) some other peer becomes interested and peer i has fewer than n_i active upload connections. The new set of upload targets includes (i) any peer currently being uploaded to, and (ii) additional peers up to the limit n_i. Using the peer selection policy, additional peers are selected from the set of interested peers that are not yet fully utilizing their download capacity. With a probability $1/n_i$ the optimistic unchoke component is used to choose a peer, and with a probability of $(n_i - 1)/n_i$ the peer that is uploading to peer i at the highest rate is chosen.

The start-up delay, that is the time since arrival until a peer begins playback, is determined using a modified version of the LTA start-up rule [14]. Playback does not commence until the following two conditions are satisfied. First, the initial two segments of the video must be fully received. Second, the measured long-term average rate at which the peer has received in-order pieces must be sufficiently high such that all of the remaining pieces would be received by their playback time, should this rate be maintained. The long-term average rate is calculated as the ratio of the amount of in-order data received thus far, divided by the time since the peer first began download of a piece that was not selected using "rarest-first with ties broken randomly", or in the case no such piece download has begun, the time since the peer's arrival to the system.[7]

For simulating the transmission rates of piece transfers, it is assumed that connection bottlenecks are located at the end points and the network operates using max-min fair bandwidth sharing (using TCP, for example). Under these assumptions, each piece transfer operates at the highest possible rate that ensures that (i) no bottleneck operates above its capacity, and (ii) the rate of no transfer can be increased without decreasing the rate of some other transfer operating at the same or lower rate.

Unless stated otherwise, it is assumed that peers have three times higher download bandwidth than upload bandwidth, and that each peer concurrently uploads to at most four peers. The maximum number of server upload connections is chosen as the total server bandwidth available for the video file divided by the video playback bit rate (an integer value owing to the parameters chosen

[6] Note that the default parameters in recent versions of the mainline BitTorrent client allow peers to be connected to up to 80 other peers, which is often achieved in practice [19]. Furthermore, peers not satisfied with their performance are able to request additional peers from the tracker.

[7] Alternative start-up rules were tried, but did not impact the relative performance of the considered policies. More aggressive rules, of course, result in shorter start-up delays but increased likelihood that a piece is not received by its playback point.

in our experiments). By ensuring that each server connection can transfer data at the playback bit rate, the server is better able to assist "low buffer" peers.

A variety of workload scenarios are considered. For scenarios with a constant-rate request (peer) arrival process, the system is simulated for 6000 requests, with the initial 1000 and the last 500 requests removed from the measurements. For flash crowd scenarios (in which the arrival rate starts high and decays to zero) no warmup period was used and simulations were run until the system emptied. Except in two scenarios considered in Section 6.6, it is assumed that all peers leave the system as soon as they have received the entire file (i.e., act only as leechers).

6 Performance Comparisons

In this section, we compare the performance of the policies defined in Sections 3 and 4. Section 6.1 describes the metrics that will be considered in the policy evaluations. Sections 6.2-6.5 present our principal comparisons for four different workload scenarios. Section 6.6 explores the impact of differing assumptions regarding the available upload resources.

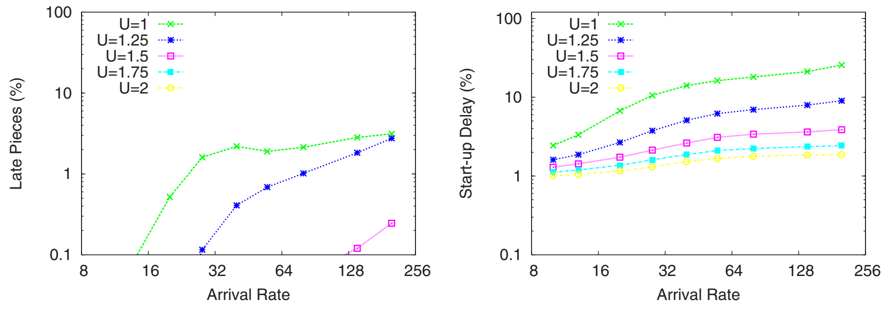
6.1 Performance Metrics

We use two quality of service metrics: (i) the *percentage of late pieces*, defined as the percentage of pieces that are not received by their playback point, and (ii) the *average start-up delay*, as determined by the start-up rule of Section 5. Without loss of generality, data volume is measured in units of the file size, and time in units of the total video playback duration. Hence, all data rates are expressed relative to the playback bit rate, and start-up delay is expressed relative to the time it takes to play the entire video. For example, an upload bandwidth of 1.25 means that when the peer is fully utilizing its upload bandwidth, it can upload data at 1.25 times the playback bit rate. Similarly, a start-up delay of 5% means that the delay until playback begins is equal to 5% of the total playback duration, and an arrival rate of 100 means that on average 100 peers arrive during the time it takes to entirely play back the video once.

6.2 Steady State Scenario

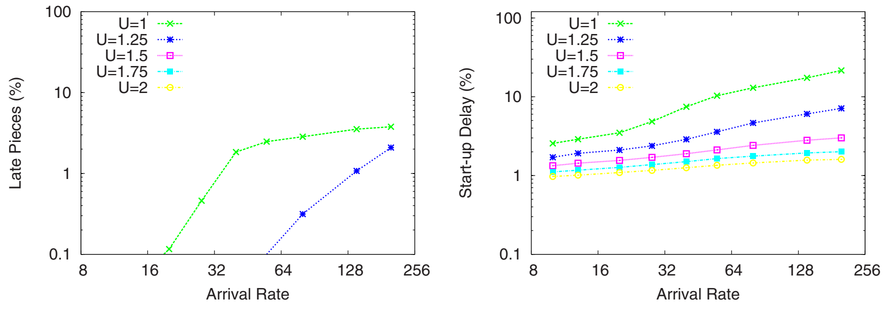
In the "steady state" scenario, peers (i) do not leave the system until having fully downloaded the file, (ii) arrive according to a Poisson process at rate λ, and (iii) are homogeneous (i.e., all peers have the same upload bandwidth U and download bandwidth D). The server upload bandwidth is denoted by B.

Figure 1 shows results for the *baseline* protocol. Figures 2 and 3 show results using RPNP and UP/RPNP, respectively. When interpreting these results, it should be noted that the total bandwidth requirement (request arrival rate × file size) ranges from two to forty times the server upload bandwidth, as the request rate varies from 10 to 200. (Naturally, at least the difference between the total bandwidth requirement and the server upload bandwidth must be contributed



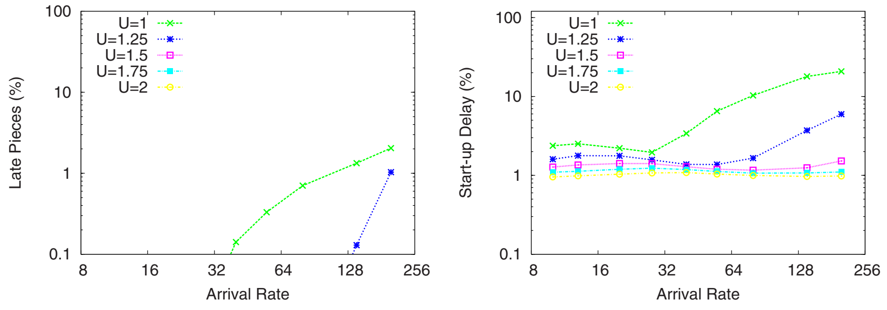
(a) Late pieces (b) Start-up delay

Fig. 1. Baseline; steady state scenario ($B = 5, D/U = 3$, 100 segments with 5 pieces each)



(a) Late pieces (b) Start-up delay

Fig. 2. Rare piece delivery to new peers (RPNP); steady state scenario ($B = 5, D/U = 3$, 100 segments with 5 pieces each)



(a) Late pieces (b) Start-up delay

Fig. 3. Urgent piece prioritization with rare piece delivery to new peers (UP/RPNP); steady state scenario ($B = 5, D/U = 3$, 100 segments with 5 pieces each)

by peers.) To capture a wide range of workloads, start-up delays, and percentage of late pieces, these figures (as well as subsequent figures) use log scales. Note that late piece percentages under 0.1% are not shown.

Comparing Figures 1 through 3, we note that RPNP substantially improves over the baseline protocol, with respect to both start-up delay (owing to "new" peers being able to compete more effectively for the upload bandwidth of other peers) and the percentage of late pieces (owing to improved piece diversity in the system). UP/RPNP provides additional improvements in both of these metrics. The improvement in start-up delay with UP/RPNP, in comparison to RPNP, arises from a subtle side-effect of using in-order rather than Zipf-based piece selection for "low buffer" peers. Generally, a more in-order piece delivery will tend to degrade the system's piece diversity somewhat, leading to a greater fraction of uploads being initiated to a random interested peer (including new peers) rather than to a peer from which data is currently being downloaded.

6.3 Flash Crowd Scenario

Our second scenario is motivated by measurements of operational file sharing torrents [20]. In this scenario, peers are assumed to arrive at an exponentially decaying rate $\lambda(t) = \lambda_0 e^{-\gamma t}$, where λ_0 is the initial arrival rate at time zero and γ is a decay factor. By varying γ between 0 and ∞, a variety of arrival processes can be simulated, from constant-rate arrivals at one extreme, to a flash crowd in which all peers arrive instantaneously (to an empty system) at the other extreme.

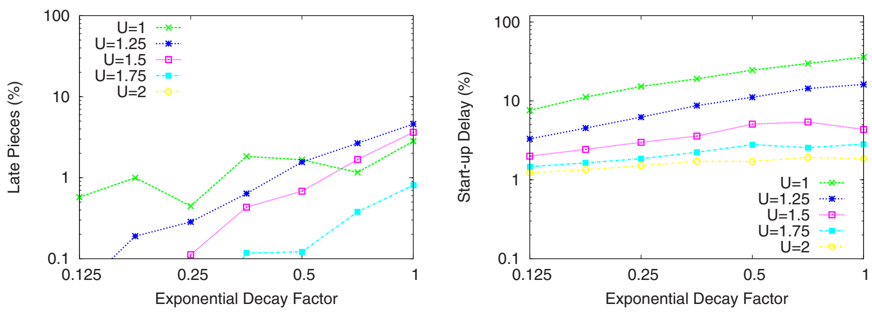
Figures 4 and 5 show the results for the second scenario using the baseline protocol and UP/RPNP, respectively. For this workload scenario, as well as for subsequent workload scenarios, results for RPNP are omitted owing to space limitations. Here, λ_0 and γ are selected such that the expected total number of arrivals is equal to 500. By varying γ between $\frac{1}{8}$ and 1, we cover a wide range of intensities of flash crowds. With $\gamma = \frac{1}{8}$, 11.8% of all arrivals occur within one playback duration of the first peer arrival; with $\gamma = 1$, the corresponding value is 63.2%.

While the potentially very high initial arrival rate makes this scenario much different than the steady state scenario, UP/RPNP still achieves significant improvements in the percentage of late pieces. The cases with a high percentage of late pieces for both protocols, which occur for intense flash crowds, are due to pieces not being disseminated from the server to all peers quickly enough. To further reduce the percentage of late pieces in these cases (given the same server resources), a more conservative start-up rule would be needed.

6.4 Heterogeneous Scenario

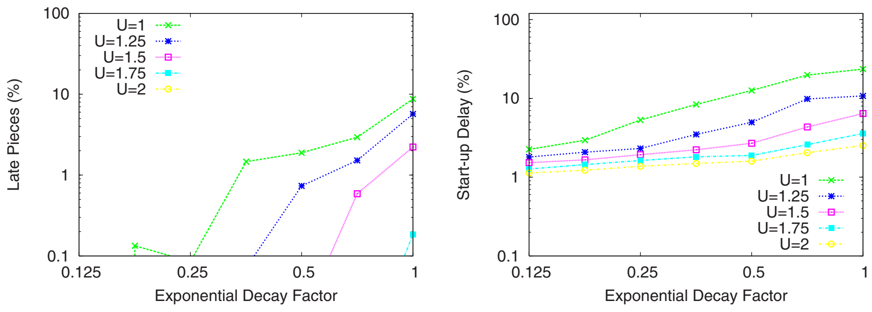
The third scenario is of a heterogeneous workload with two types of peers: low bandwidth peers and high bandwidth peers. For both types of peers, the download bandwidth is three times the upload bandwidth, as assumed previously. As in the first scenario peers arrive at a constant rate λ.

Figure 6 shows the percentage of late pieces and the average start-up delay for this workload scenario. The percentage of high bandwidth peers is varied such that the system ranges from bandwidth-constrained (most peers are low bandwidth) to bandwidth-rich (most peers are high bandwidth). Results are shown for a workload in which the low and high bandwidth peers have upload



(a) Late pieces (b) Start-up delay

Fig. 4. Baseline; flash crowd scenario ($B = 10, D/U = 3$, 100 segments with 5 pieces each)



(a) Late pieces (b) Start-up delay

Fig. 5. Urgent piece prioritization with rare piece delivery to new peers (UP/RPNP); flash crowd scenario ($B = 10, D/U = 3$, 100 segments with 5 pieces each)

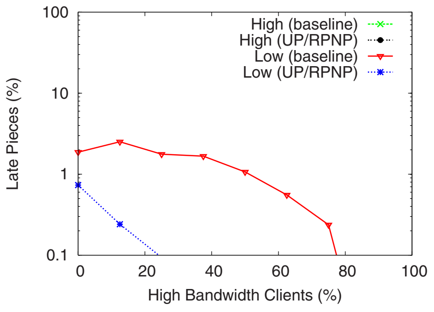
bandwidths of one and two times the playback bit rate, respectively. As in the previous scenarios, significant improvements in both the percentage of late pieces and the start-up delays are observed with UP/RPNP.

Note that for both the baseline and UP/RPNP, the high bandwidth peers generally incur both a smaller percentage of late pieces and lower start-up delays. This is a consequence of both the higher download bandwidth of these peers, and of the use of tit-for-tat coupled with their higher upload bandwidth.

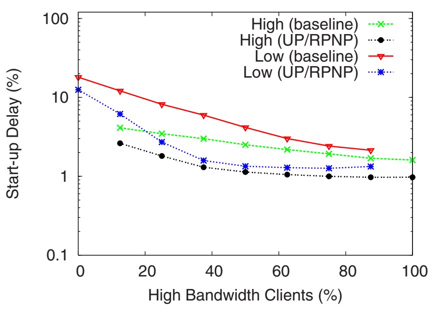
6.5 Freeloader Scenario

Our fourth scenario is of a workload with two types of peers: contributing peers and freeloaders. We assume that both types of peers have identical download bandwidth, equal to three times their upload bandwidth, as assumed previously. However, only the contributing peers upload file pieces to other peers.

Figure 7 shows the percentage of late pieces and the average start-up delay for this workload scenario. Results are shown for a workload in which peers have an upload bandwidth of 1.25 times the playback bit rate. The percentage of

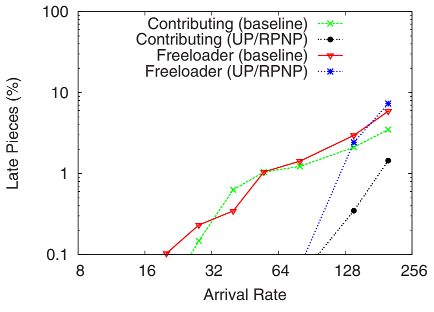


(a) Late pieces

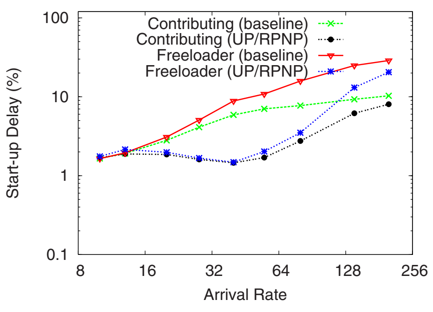


(b) Start-up delay

Fig. 6. Impact of heterogeneity; steady state ($B = 5, \lambda = 100, D/U = 3$, $U_{high} = 2$, $U_{low} = 1$, 100 segments with 5 pieces each)



(a) Late pieces



(b) Start-up delay

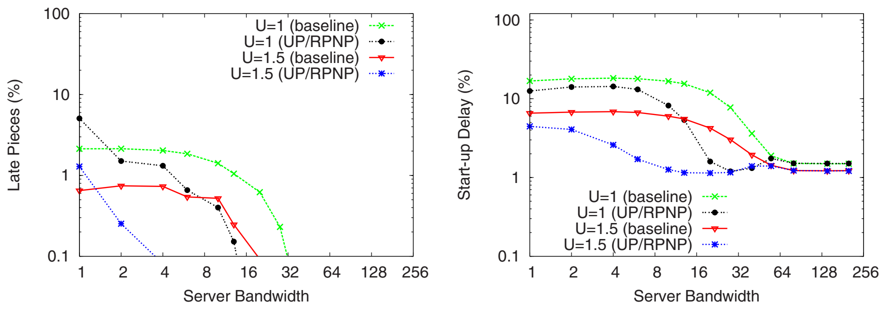
Fig. 7. Performance with freeloaders; steady state ($B = 5, D = 3.75$, $U = 1.25$, 5% freeloaders, 100 segments with 5 pieces each)

peers that are freeloaders is fixed at 5%, and the arrival rate is varied between 10 and 200. As in the previous scenarios, significant improvements in both the percentage of late pieces and the start-up delays are observed with UP/RPNP.

With both the baseline protocol and UP/RPNP, the rate-based tit-for-tat mechanism ensures that contributing peers receive substantially better performance than the freeloaders. For example, with UP/RPNP and $\lambda = 200$, freeloaders have an average start-up delay roughly three times that of contributing peers, and observe more than five times as many late pieces. While freeloaders are regularly unchoked (due to the use of optimistic unchoke), and therefore, are not completely starved, this performance advantage illustrates that the tit-for-tat policy provides peers with a strong incentive to contribute their upload resources.

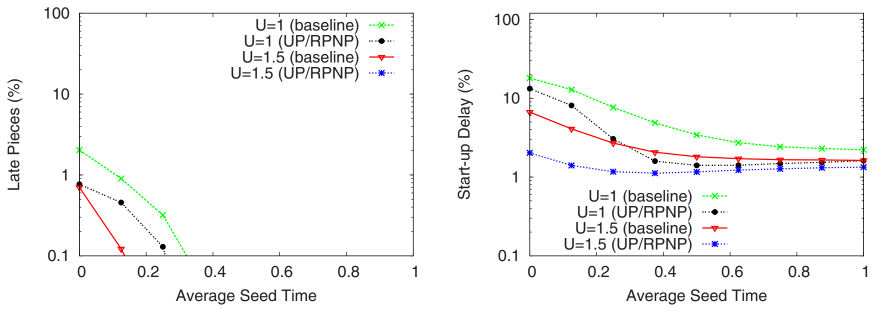
6.6 Impact of Total Upload Capacity

Figures 8 through 10 show the percentage of late pieces and the average start-up delay as functions of the server upload bandwidth, the average time peers stay in the system after having completed download as "seeders", and the peer arrival



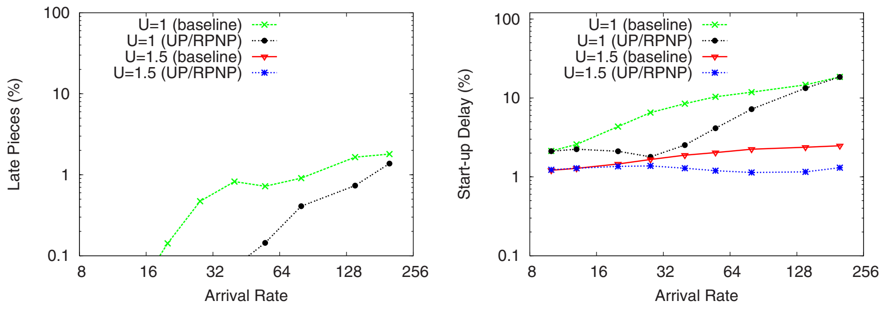
(a) Late pieces (b) Start-up delay

Fig. 8. Impact of server bandwidth; steady state scenario ($\lambda = 100, D/U = 3$, 100 segments with 5 pieces each)



(a) Late pieces (b) Start-up delay

Fig. 9. Impact of the average seed time; steady state scenario ($B = 5, \lambda = 100, D/U = 3$, 100 segments with 5 pieces each)



(a) Late pieces (b) Start-up delay

Fig. 10. Steady state scenario in which peers stay until playback completion ($B = 5, D/U = 3$, 100 segments with 5 pieces each)

rate for an example scenario in which peers stay in the system until having played back the entire file (rather than only until download completion), respectively. For simplicity, we assume that seed times are exponentially distributed.

As expected, increases in the server bandwidth (cf. Figure 8) and the available peer upload bandwidth (cf. Figures 9 and 10) have positive impacts on the quality of service. Comparing Figure 10 with Figures 1 and 3, we note that some performance improvements are possible if peers stay in the system contributing their upload bandwidth at least until having completed playback (rather than just until download is complete).

7 Conclusions

This paper has considered the problem of devising BitTorrent-like peer-assisted protocols for on-demand video streaming systems, in which peers are motivated to upload data to others owing to the likely beneficial impact on their own achieved performance. The challenge in this context is that of mediating the conflict between the goals of low start-up delay and consistently on-time piece delivery (which motivates piece delivery that is more "in-order"), and the requirements of effective tit-for-tat (which motivates piece delivery that is more "rarest first").

We devised new tit-for-tat compatible policies where the server, for which tit-for-tat is not an issue, gives preference to peers at imminent risk of receiving data too late for playback, and secondly to upload of rare pieces to newly arrived peers. Evaluations of the proposed policies for a variety of workload scenarios suggests that our policies are able to provide substantial improvements in quality of service while ensuring that the piece diversity is sufficient for peers to effectively employ tit-for-tat.

References

1. Cohen, B.: Incentives Build Robustness in BitTorrent. In: Proc. Workshop on Economics of Peer-to-Peer Systems 2003, Berkeley, CA (June 2003)
2. Huang, C., Li, J., Ross, K.W.: Can Internet Video-on-Demand be Profitable? In: Proc. ACM SIGCOMM 2007, Kyoto, Japan, pp. 133–144 (August 2007)
3. Janardhan, V., Schulzrinne, H.: Peer Assisted VoD for Set-top Box Based IP Network. In: Proc. ACM SIGCOMM Workshops (Peer-to-Peer Streaming and IP-TV) 2007, Kyoto, Japan (August 2007)
4. Cui, Y., Li, B., Nahrstedt, K.: ostream: Asynchronous streaming multicast in application-layer overlay networks. IEEE JSAC 22(1), 91–106 (2004)
5. Sharma, A., Bestavros, A., Matta, I.: dPAM: A Distributed Prefetching Protocol for Scalable Asynchronous Multicast in P2P Systems. In: Proc. IEEE INFOCOM 2005, Miami, FL, pp. 1139–1150 (March 2005)
6. Hefeeda, M., Habib, A., Botev, B., Xu, D., Bhargava, B.: PROMISE: Peer-to-Peer Media Streaming using CollectCast. In: Proc. ACM MM 2003, Berkeley, CA, pp. 45–54 (November 2003)
7. Rejaie, R., Ortega, A.: PALS: Peer-to-Peer Adaptive Layered Streaming. In: Proc. NOSSDAV 2003, Monterey, CA, pp. 153–161 (June 2003)
8. Vratonjic, N., Gupta, P., Knezevic, N., Kostic, D., Rowstron, A.: Enabling DVD-like Features in P2P Video-on-demand Systems. In: Proc. ACM SIGCOMM Workshops (Peer-to-Peer Streaming and IP-TV) 2007, Kyoto, Japan (August 2007)

9. Parvez, N., Williamson, C., Mahanti, A., Carlsson, N.: Analysis of BitTorrent-like Protocols for On-demand Stored Media Streaming. In: Proc. ACM SIGMETRICS 2008, Annapolis, MD, pp. 301–312 (June 2008)
10. Cheng, B., Stein, L., Jin, H., Zhang, Z.: Towards Cinematic Internet Video-on-Demand. In: Proc. EuroSys 2008, Glasgow, Scotland, pp. 109–122 (March 2008)
11. Mol, J.J.D., Pouwelse, J.A., Meulpolder, M., Epema, D.H.J., Sips, H.J.: Give-to-Get: Free-riding Resilient Video-on-Demand in P2P Systems. In: Proc. MMCN 2008, San Jose, CA (January 2008)
12. Annapureddy, S., Guha, S., Gkantsidis, C., Gunawardena, D., Rodriguez, P.R.: Is High-Quality VoD Feasible using P2P Swarming? In: Proc. WWW 2007, Banff, Canada, pp. 903–912 (May 2007)
13. Vlavianos, A., Iliofotou, M., Faloutsos, M.: BiToS: Enhancing BitTorrent for Supporting Streaming Applications. In: Proc. Global Internet Workshop 2006, Barcelona, Spain (April 2006)
14. Carlsson, N., Eager, D.L.: Peer-assisted On-demand Streaming of Stored Media using BitTorrent-like Protocols. In: Akyildiz, I.F., Sivakumar, R., Ekici, E., de Oliveira, J.C., McNair, J. (eds.) NETWORKING 2007. LNCS, vol. 4479, pp. 570–581. Springer, Heidelberg (2007)
15. Choe, Y.R., Schuff, D.L., Dyaberi, J.M., Pai, V.S.: Improving VoD Server Efficiency with BitTorrent. In: ACM MM 2007, Augsburg, Germany, pp. 117–126 (September 2007)
16. Garbacki, P., Epema, D.H.J., Pouwelse, J., van Steen, M.: Offloading Servers with Collaborative Video on Demand. In: Proc. IPTPS 2008, Tampa Bay, FL (February 2008)
17. Gkantsidis, C., Rodriguez, P.R.: Network Coding for Large Scale Content Distribution. In: Proc. IEEE INFOCOM 2005, Miami, FL, pp. 2235–2245 (March 2005)
18. Zhang, X., Liu, J., Li, B., Yum, T.-S.P.: CoolStreaming/DONet: A Data-driven Overlay Network for Peer-to-Peer Live Media Streaming. In: Proc. IEEE INFOCOM 2005, Miami, FL, pp. 2102–2111 (March 2005)
19. Legout, A., Urvoy-Keller, G., Michiardi, P.: Rarest First and Choke Algorithms Are Enough. In: Proc. ACM IMC 2006, Rio de Janeiro, Brazil, pp. 203–216 (October 2006)
20. Guo, L., Chen, S., Xiao, Z., Tan, E., Ding, X., Zhang, X.: Measurement, Analysis, and Modeling of BitTorrent-like Systems. In: Proc. ACM IMC 2005, Berkley, CA, pp. 35–48 (October 2005)