Peer Selection Strategies for Improved QoS in Heterogeneous BitTorrent-like VoD Systems

Lucia D'Acunto, Nazareno Andrade, Johan Pouwelse and Henk Sips
Delft University of Technology, The Netherlands
l.dacunto@tudelft.nl

***Abstract*—The efficiency of BitTorrent in disseminating content has inspired a number of P2P protocols for on-demand video streaming (VoD). Prior work on adapting BitTorrent to VoD mainly focused on the piece selection policy, since streaming requires a somewhat "in order" download progress. Conversely, not much effort has been spent into adapting BitTorrent's peer selection policy, where nodes mainly serve those that have recently uploaded to them at the highest rates. This mechanism incentivizes cooperation among peers but, in a heterogeneous system (i.e. where peers have different bandwidth capacities), it causes faster peers to receive higher download speeds than slower peers. This might hurt the system's ability of providing as many nodes as possible with the minimum download speed necessary to sustain the video playback rate. Furthermore, peers gain little utility in downloading at rates much higher than the video playback rate. Inspired by these observations, in this work, we extend the peer selection mechanism of an existing BitTorrent-like VoD protocol, *give-to-get* (G2G), with techniques that allow peers to relax their reciprocity-based peer selection and choose more random nodes when their current QoS is high. In this way, more peers can be granted a good QoS and free-riding is tolerated only when bandwidth resources are abundant. To demonstrate the benefits of our approach, we present extensive simulations of the introduced techniques.**

I. Introduction

In recent years, significant research effort has focused on how to efficiently use a peer-to-peer (P2P) architecture to provide large-scale video-on-demand (VoD) services [23], [6], [13], [16], [22]. In particular, much has been investigated on how to adapt the design of BitTorrent to produce robust and efficient P2P VoD protocols [4], [18], [17], [11], [21], [13]. In the context of traditional file transfer, BitTorrent combines a high utilization of peers' upload bandwidth [7] and incentives for cooperation, which are two attractive properties for VoD systems as well. However, BitTorrent was not designed to address two central requirements for VoD systems: (i) *startup delay minimization*: the time a user needs to wait for the video playback to start must be short; and (ii) *stream continuity preservation*: once the playback has started, pieces must be downloaded before a certain deadline to avoid stall times and missed frames.

Two main components of the BitTorrent protocol can be modified to address the VoD requirements: the piece selection policy, and the peer selection policy. BitTorrent's piece selection policy, Local Rarest First (LRF), maximizes piece availability but does not cater for the in-order download necessary to preserve stream continuity and shorten startup delays. Hence, considerable effort on adapting BitTorrent to VoD has focused on changing this policy and finding a compromise between in-order download and high upload capacity utilization [4], [17], [18], [21], [13].

The problem of adapting BitTorrent's peer selection policy, conversely, has not received as much attention from the research community. With BitTorrent's policy, tit-for-tat (T4T), nodes employ *direct reciprocity* [15] and prefer uploading to the peers that have recently contributed to them at the highest speeds. The emergent effect of this policy is that peers having higher upload capacities typically have higher download speeds [3], and hence incentives for cooperation. Although these are desirable properties for the scenario of traditional file transfer, this policy has two drawbacks for VoD:

a) direct reciprocity is less feasible in VoD systems, since it is difficult for younger peers to reciprocate older peers: nodes arrived later have made fewer progress on their downloads and have a playback position behind that of peers who have been in the system for longer; as a result, using T4T in VoD does not produce the same incentives as in traditional file transfer.
b) T4T was designed to maximize the download speed of users, while in VoD systems users gain little utility from having a download speed higher than the necessary to preserve stream continuity. The use of T4T in a VoD system with heterogeneous peers can lead to situations where, even if there is enough aggregate upload capacity to serve all peers in the system, not all peers experience a satisfactory QoS. This happens because high-capacity peers may receive download rates substantially higher than the video playback rate, while low-capacity peers experience download rates that are too low to meet the VoD requirements.

Give-to-get (G2G) [11] is a BitTorrent-based P2P VoD protocol that adapts the peer selection policy trying to preserve T4T's incentives for cooperation. Because of the difficulty of younger peers to reciprocate older peers, in G2G a node prefers uploading to peers that have recently *forwarded* pieces to others at the highest rate. G2G therefore uses *indirect reciprocity* [15] to provide incentives in the VoD context. It does not, however, cater for issue (b), as the reward given to peers is similar to that of T4T. It follows that heterogeneity can still lead to poor QoS for some peers even in the presence of enough capacity to serve all.

In this work, we focus on adapting the peer selection

978-0-7695-4217-1/10 $26.00 © 2010 IEEE
DOI 10.1109/ISM.2010.22

policy of G2G to maximize the number of peers that are able to meet the QoS requirements needed for streaming, while maintaining G2G's current incentives. Our approach is to let nodes act more altruistically if they are receiving a service that is higher than the necessary to preserve stream continuity. More specifically, our contributions are as follows:

1) we propose an adaptive strategy for peers to decide, in relation to their current progress, whether relaxing the reciprocity-based peer selection and serve more random peers (Section III); in this way, when bandwidth is abundant, a larger fraction of peers can achieve a satisfactory QoS, in terms of both startup delay and stream continuity;
2) we extend this adaptive mechanism to have nodes give preference to newly joined peers (Section III); by doing so, the average startup delay can be reduced even further, with little or no negative effect on the stream continuity;

Simulation results show that, in heterogeneous systems where bandwidth is abundant, our adaptive peer selection policy significantly increases the number of low-capacity peers receiving a satisfactory QoS (up to 4 times), without affecting that of high-capacity peers. When extended to give preference to newcomers, this strategy can reduce startup delays (up to 48% in the scenarios considered) with no significant impact on the stream continuity. On the other hand, if bandwidth is scarce, higher-contributors are prioritized (Section V).

II. Background

In this section, we first provide an overview of both BitTorrent and G2G and then we review previous work on adapting BitTorrent to VoD.

A. BitTorrent

BitTorrent is a widely popular P2P protocol for content distribution. In BitTorrent, files are split into pieces, allowing peers which are still downloading content to serve the pieces they already have to others. Each node establishes persistent connections with a large set of peers (typically between 40 and 80), called its *neighborhood*, and uploads data to a subset of this neighborhood.

More specifically, each peer divides equally its upload capacity into a number of *upload slots*. The decision to allocate an upload slot to a peer is termed an *unchoke* in BitTorrent, and there are two types of upload slots: *regular unchoke slots* and *optimistic unchoke slots*. Regular unchoke slots are assigned according to the T4T policy: peers prefer other nodes that have recently provided data at the highest speeds. Each peer re-evaluates the allocation of its regular unchoke slots every *unchoke interval* δ (generally 10 seconds). Different from the regular unchoke slots, the optimistic unchoke ones are assigned to randomly selected nodes. Also, their allocation is re-evaluated every *optimistic unchoke interval*, which is generally set to 3δ. Optimistic unchoke slots serve the purposes of (i) having peers discover new, potentially faster, nodes to unchoke so as to be reciprocated, and (ii) bootstrapping newcomers (i.e. peers with no pieces yet) in the system. Peers that are currently assigned an upload slot (whether regular or optimistic) from a node p are said to be *unchoked* by p; all the others are said to be *choked* by p.

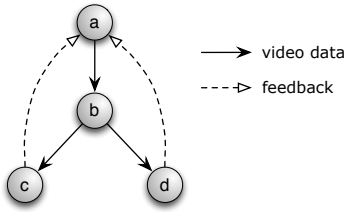

Fig. 1. Data flow and feedback flow in G2G.

Each peer maintains its neighborhood informed about the pieces it owns. The information received from its neighborhood is used to request pieces of the file according to the Local Rarest First (LRF) policy. This policy determines that each peer requests the pieces that are the rarest among its neighbors. The emergent effect of this policy is that less-available pieces get replicated fast among peers and each peer obtains first the pieces that are most likely to interest its neighbors [2].

B. Give-to-Get (G2G)

Give-to-Get [11] is a P2P VoD protocol which is inspired by BitTorrent and is currently implemented in the Tribler P2P system [1], [20]. Its efficiency has been shown in simulations as well as in public trials with real users [10]. G2G modifies both the piece selection policy and the peer selection policy of BitTorrent. With respect to the piece selection policy, in G2G peers prioritize the download of pieces that are close to the current playback position. Pieces that can not be downloaded before their playback deadline are considered as missed. For what concerns peer selection, G2G uses regular and optimistic unchoke slots just as BitTorrent. For the regular unchoke slots, each peer select the nodes that have forwarded pieces received from it at the highest speeds. In order to discover the forwarding rate of a child node b, a peer a periodically asks its grandchildren about the pieces received from b. (the child b is not asked directly as it could make false claims). This process is depicted in Figure 1. The optimistic unchoke slots are assigned in a similar way as in BitTorrent and the optimistic unchoke interval is 2δ.

Although outside the scope of this work, we observe that a protocol where peers rely on indirect information is potentially more vulnerable to attacks, such as the creation of many synthetic identities, the so-called sybil attack [8], and collusion. The sybil attack is especially problematic, since a peer p can create many fake children which will report p is forwarding to them at a high rate. This type of attack can, however, be prevented by means of techniques used by other P2P protocols for inhibiting the creation of multiple identities [8], [12]. For example, the creation of new identities can be limited on the basis of durable identifiers (in this case, a subscription to the system is needed). Furthermore, Piatek et al. [12] show

a number of techniques that can be used to limit the effects of collusion in P2P systems that rely on indirect information.

C. Related Work on Adapting BitTorrent to VoD

Prior work on adapting BitTorrent to VoD mainly addresses the piece selection policy (see [4], [11], [17], [18], [21], [13]). These studies focus on finding a policy that can achieve a good trade-off between the need of sequential download progress and high piece diversity. Although the approaches proposed to tackle this problem may vary, they all have in common that the resulting strategy combines in-order piece selection with rarest-first piece selection. By doing so, they succeed in reaching a good compromise between sequential download progress and high utilization of peer upload capacity.

The problem of adapting the peer selection policy has received less attention. Shah et al. [18] propose a strategy where peers perform a new optimistic unchoke round everytime a piece is played. We observe two issues with this approach: (i) frequent unchoke rounds might weaken incentives for cooperation, (ii) a peer can compromise its own QoS by being too altruistic. In contrast, with our proposed strategies, a peer will behave more altruistically only when it is receiving a service that is substantially better than the necessary to preserve QoS. A different kind of peer selection problem is analyzed by Yang et al. in [22]. Specifically, they study a number of strategies for a peer to choose which other node a request should be sent to, in order to balance the load among requested nodes and increase the likelihood of receiving the needed pieces before their deadlines.

With respect to reducing startup delays, Carlsson et al. [14] propose a policy where newly joined peers are given the rarest pieces. In this way, once they have obtained their first piece, these nodes become attractive bartering partners for other peers and are likely to enjoy fast download speed sooner. Instead, as we will see in Section III, we modify the peer selection mechanism in order to reduce the time needed by newcomers to receive the first piece.

III. Proposed Peer Selection Strategies

In this section, we present our proposed strategies to improve the QoS of peers in a heterogeneous environment.

A. Proportional slot number

The first step towards our adaptive strategies consists into adjusting the number of upload slots a peer opens.

In the original G2G protocol, the number of upload slots is calculated as follows. One slot is reserved for optimistic unchoke. Other three are reserved for regular unchokes and, if there is more spare capacity (e.g. when the already selected receiving peers hold download bottlenecks), a peer will open a maximum of 2 more regular unchoke slots.

However, with this rule, it might happen that a high-capacity peer will provide each of its (few) children with a very high bandwidth, while some other nodes in the system are experiencing insufficient speeds to achieve reasonable QoS. Since peers do not earn more utility in downloading at rates substantially higher than the video playback rate, and balancing the service each peer receives is desirable, it makes more sense that a node limits the bandwidth provided to each child and opens a number of upload slots according to its total upload bandwidth, as well as the video playback rate. We propose the following rule to calculate how many upload slots s a peer p opens:

$$s = \max\left(s_0, \left\lceil s_0 \frac{\mu}{R} \right\rceil\right), \tag{1}$$

where μ is p's upload capacity, R is the video playback rate and s_0 is a protocol parameter which represents the minimum number of upload slots each peer opens. With this rule, a high-capacity peer ($\mu > R$) will try to provide each of its children with an upload capacity of $\sim \frac{R}{s_0}$. In this way, no peer will receive from another peer pieces at a rate higher than R (unless the uploading peer has some more capacity left, i.e. if the other nodes it is uploading to hold download bottlenecks) and the bandwidth of high-capacity nodes can be shared among more peers.

Given a number of slots, it is necessary also to determine how many of them are reserved for optimistic unchoke. Leaving only one slot for this purpose means that, the larger a peer's bandwidth, the smaller the fraction of it dedicated to optimistic unchoke. Jia et al. [5] show that doing so leads to a larger service difference among heterogeneous peers. To avoid that, nodes should reserve always the same fraction of their bandwidth to the optimistic unchoke. Hence, we propose that each peer is initialized with o_{min} optimistic unchoke slots where o_{min} is:

$$o_{min} = round\left(\frac{s}{s_0}\right), \tag{2}$$

in this way, each peer will assign $\sim \frac{1}{s_0}$ of its upload capacity to optimistic unchoke.

In the strategies that follow, we assume that a peer's upload slot number is always initialized according to this rule, which we call *proportional slot number rule* (PSN).

Algorithm 1 Adapting optimistic unchoke slot number to sequential progress

```
t := time now
w := window length
s := total number of upload slots
o := number of optimistic unchoke slots
o_min ⇐ round(s/s_0)
for each optimistic unchoke round do
    if sequential progress(t,w) > T_up then
        o ⇐ min(s, o+1)
    end if
    if sequential progress(t,w) < T_lw then
        o ⇐ max(o_min, o−1)
    end if
end for
```

B. Adaptive optimistic unchoke slot number

The core of this strategy is to have peers dynamically adjust the number of their optimistic unchoke slots to their current

Algorithm 2 Optimistic unchoke algorithm with priority to newcomers (PNC)

```
1: s := total number of upload slots
2: o := number of optimistic unchoke slots
3: c ⇐ number of newcomers that can be preferentially unchoked
4: P ⇐ list of all choked neighbors having no pieces yet
5: o' ⇐ max(o, s − number of already unchoked neighbors)
6: for i = 1 to min(o', c) do /* first, unchoke the newcomers */
7:    unchoke(P[i])
8: end for
9: N ⇐ all choked interested neighbors
10: if (o' − c) > 0 then /* if there are some slots left, then */
11:    for i = 1 to (o' − c) do /* unchoke some other peers */
12:       unchoke(N[i])
13:       put N[i] at the bottom of the list /* so that at the next
          optimistic unchoke round another peer is chosen */
14:    end for
15: end if
```

QoS. To measure the QoS a peer is currently receiving, we use the *sequential progress*, defined as the speed at which the index of the first missing piece in the file grows. This measure is an indicator for the preservation of the continuity of the stream: a sequential progress faster than the video playback rate implies a continuous stream. At the same time, this metric is agnostic with respect to the underlying piece selection policy, making a strategy based on it directly portable to other P2P VoD protocols.

Since a peer p that is already receiving a high QoS does not earn more utility in having a sequential progress much higher that the video playback rate R, we propose that, in such a situation, p will decide to "help" other peers by increasing the number of its optimistic unchoke slots. On the other hand, it will reduce its optimistic unchoke slots if its sequential progress has decreased past a limit. To avoid switching too often between increasing and decreasing the number of optimistic unchoke slots, we use the following approach. First, we define an interval (T_{lw}, T_{up}) for the equilibrium value of the sequential progress. A peer p will then increase the number of its optimistic unchoke slots only when its sequential progress is above the upper threshold T_{up} and decrease it when its sequential progress goes below the lower threshold T_{lw}. Second, at every optimistic unchoke round, the current sequential progress is calculated over the last w seconds (Algorithm 1). By opportunely tuning the thresholds T_{lw} and T_{up} and the length of the window w, the performance of other peers can improve, while that of peer p will stay constant. In particular, in order to guarantee that peer p does not get worse off from being too altruistic, for T_{lw} it is advisable a value larger than R. Similarly, the size of the window w should be large enough to capture the trend of a peer's sequential progress, besides eventual instantaneous oscillations, but small enough to be sensitive to a change of trend.

C. Priority to newcomers in optimistic unchoke slots

When a new peer joins the network, it needs to wait for a certain period of time to be optimistically unchoked by other peers. Once a node has obtained the first piece, it can forward it to other peers and get a chance of being selected by its parents in their regular unchoke slots. We denote this time in which peers are waiting for the first piece as "bootstrap time". It has to be noticed that peers are subjected to a bootstrap time also in a regular BitTorrent system. However, compared to the total download time that nodes have to wait before they can access the file, the bootstrap time becomes insignificant. On the contrary, in a VoD system, where playback starts well before the file is completely downloaded, the bootstrap time can represent an important fraction of the startup delay [21].

Hence, the startup delay can be reduced by decreasing the bootstrap time. In order to do so, we propose to extend the previous strategy with a mechanism where nodes give *priority to newcomers* when performing optimistic unchoke. In this way, newcomers will not need to wait too long before being unchoked for the first time. Algorithm 2 illustrates the pseudo-code for this strategy. When deciding how many newcomers will be given priority to, there are two things to consider: the new strategy should not (i) disrupt one of the original purpose of optimistic unchoke slots, i.e. finding new, potentially better, neighbors, nor (ii) neutralize the benefits of the adaptive strategy introduced above. In order to do so, we propose a *priority to newcomers strategy* (PNC) where the number c of newcomers that will be given priority is calculated as follows:

$$c = \left\lfloor \frac{o - o_{min}}{2} \right\rfloor, \tag{3}$$

where o is the current number of optimistic unchoke slots opened by a peer p.

Hence, when p has only o_{min} optimistic unchoke slots, it will assign them to randomly selected peers, in this way we eliminate the risk that a node gets worse off from helping newcomers. A node p will favor newcomers only when it has at least $o_{min} + 2$ optimistic unchoke slots (meaning that its current QoS is high) and will use at most half of them, so that there are enough left for helping regular peers as well.

To prevent nodes from lying about being newcomers, a peer will only upload to newcomers pieces from an initial buffer (i.e. any piece from the first h pieces of the file).

IV. Performance Metrics and Experimental Setup

We evaluate the effects of the proposed strategies on the QoS of G2G by means of simulations.

A. Methodology

We have implemented G2G and all our strategies on top of the MSR BitTorrent simulator [7]. This discrete event-based simulator accurately emulates the behavior or BitTorrent at the level of piece transfers and has been widely used, also for simulating BitTorrent-like VoD protocols [21], [22].

The settings for our simulations are shown in Table I. We consider a system in steady state[1], which is emulated as follows. First, the system is initialized with N peers having

[1]a system is said to be in steady state when, although peers might join and leave, the total number of peers remains constant.

Parameter	Value
Number of concurrent nodes N	200
Video playback rate R	800 Kbps
Video length L	30 min
Size h of the initial buffer	20 pieces
Piece size	256 KB
Peer set size	50
Min number of upload slots s_0	5
Upload capacity of the provider	8000 Kbps ($10R$)
Simulation time	3 hours ($6L$)

TABLE I
SIMULATION SETTINGS

random level of progress. Then, as soon as a peer completes the download, it leaves the system while a new peer joins. To keep the aggregate upload capacity of the system constant, the joining peer is initialized with the same upload capacity as the peer who just left. All the nodes that joined in the first L minutes (which include the initialization nodes) and the peers whose download was not complete yet at the time the simulation ended, are discarded from our results. For the adaptive strategies, we have used $T_{lw} = 1.2R$, $T_{up} = 1.4R$ and $w = 2min$, as in our experiments these values have shown to provide a good trade-off between maximizing altruism and retaining a good QoS for the altruistic peers.

Furthermore, in our simulations we assume that (i) all peers have a download capacity five times higher than their upload capacity, (ii) peers leave as soon as they finish download, (iii) there is one peer holding a complete copy of the file, set up by the provider of the video file, which never leaves the system.

Finally, we use the LTA rule introduced in [13] to decide when playback can safely commence. Specifically, a peer will start playback only when it has obtained all the pieces in the initial buffer and its current sequential progress is such that, if maintained, the download of the file will be completed before playback ends.

B. Performance Metrics

To analyze how well the startup delay minimization requirement is met, we have measured the average and the 95th percentile of the startup delay.

For the preservation of stream continuity, a common metric is represented by the *continuity index* (CI), defined as the ratio of pieces received before their deadline over the total number of pieces. Habib et al. [9] show that when the continuity index of a stream is 0.95, a user's overall perceived video quality is very good. Similarly, they show that a CI $\leq$ 0.75 leads to a poor video quality. Hence, we evaluate the ability of the P2P VoD system to maximize the number of peers receiving a high stream continuity by characterizing the fraction of peers receiving a good service (i.e. whose CI is above 95%) and poor service (i.e. whose CI index below 75%).

C. Experiments

We compare the baseline G2G with our introduced strategies: the PSN rule alone, the adaptive policy and the adaptive policy enhanced with PNC (adaptive + PNC), under different resource supply conditions:

a) *overprovision scenario*: the average peer upload capacity is greater than the video playback rate R;
b) *contention scenario*: the average peer upload capacity is smaller than the video playback rate R.
c) *dynamic scenario*: the average peer upload capacity is initially greater than the video playback rate R and becomes smaller suddenly, in the middle of the simulation.

V. RESULTS AND ANALYSIS

In this section, we present the evaluation of our proposed strategies in light of the experimental setup described in Section IV.

A. Overprovision Scenario

In this scenario, we consider a system where 50% of the peers have high upload capacity and the other 50% have slow upload capacity. Although the bandwidths of fast and slow nodes, μ_f and μ_s respectively, are different in each setup, the average peer upload capacity is always $1.25R$. This is larger than the average demand (which is equal the playback rate R). In each setup, the difference between μ_f and μ_s is increased, in order to evaluate how the strategies perform with higher heterogeneity. The homogeneous case, where all peers have upload capacity $1.25R$, is reported as well for comparison. The details for these setups are shown in Table II.

The fraction of peers receiving good ($CI \geq 0.95$) and poor ($CI \leq 0.75$) service is depicted in Figure 2. From these graphs we can make the following observations:

- Our strategies do not affect the homogeneous setup. In this case, all peers have the same, high enough, upload capacity and do not need the altruism of other peers to get a better service.
- In the heterogeneous setups, PSN alone can already provide a better QoS than the baseline G2G. For example, in the setup with high heterogeneity, the fraction of slow peers receiving a good service increases by 150% (from 0.13 to 0.33), and that of slow peers receiving poor service decreases by 60% (from 0.48 to 0.19).
- When using the adaptive policy, the gain is even higher. In the setup with mild heterogeneity, the fraction of slow peers receiving good service increases by 43% (from 0.58 to 0.83). In the setup with high heterogeneity the fraction of slow peers receiving a good service is 4 times as high as with the baseline G2G (0.57 against 0.13).
- In all setups, both PSN and the adaptive policy do not significantly affect the service received by fast peers.
- The fraction of peers receiving poor service decreases in all setups when using PSN and the adaptive policy.

Figure 3 reports the mean and the 95th percentile of the startup delay. We can notice:

- In the homogeneous setup, adaptive + PNC can substantially reduce the startup delays (48% for the mean and 53% for the 95th percentile) without impact on the respective CIs (see Figure 2).

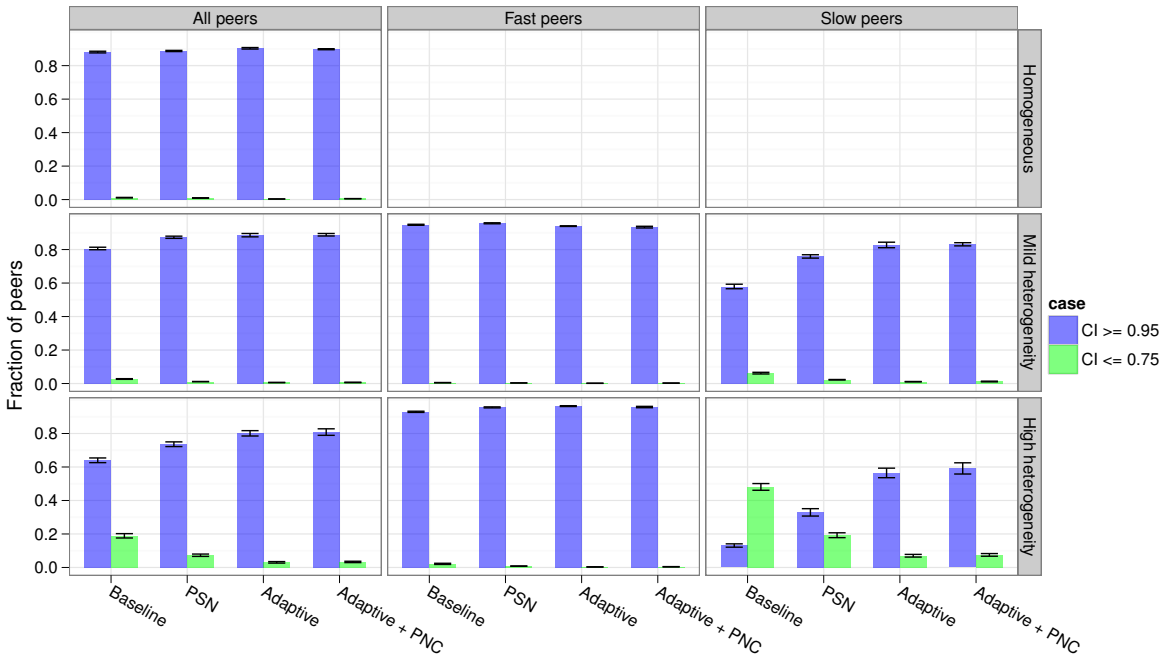

Fig. 2. Overprovision scenario: fraction of peers receiving a good service ($CI \geq 0.95$) or a poor service ($CI \leq 0.75$). The error bars correspond to a confidence level of 95%.

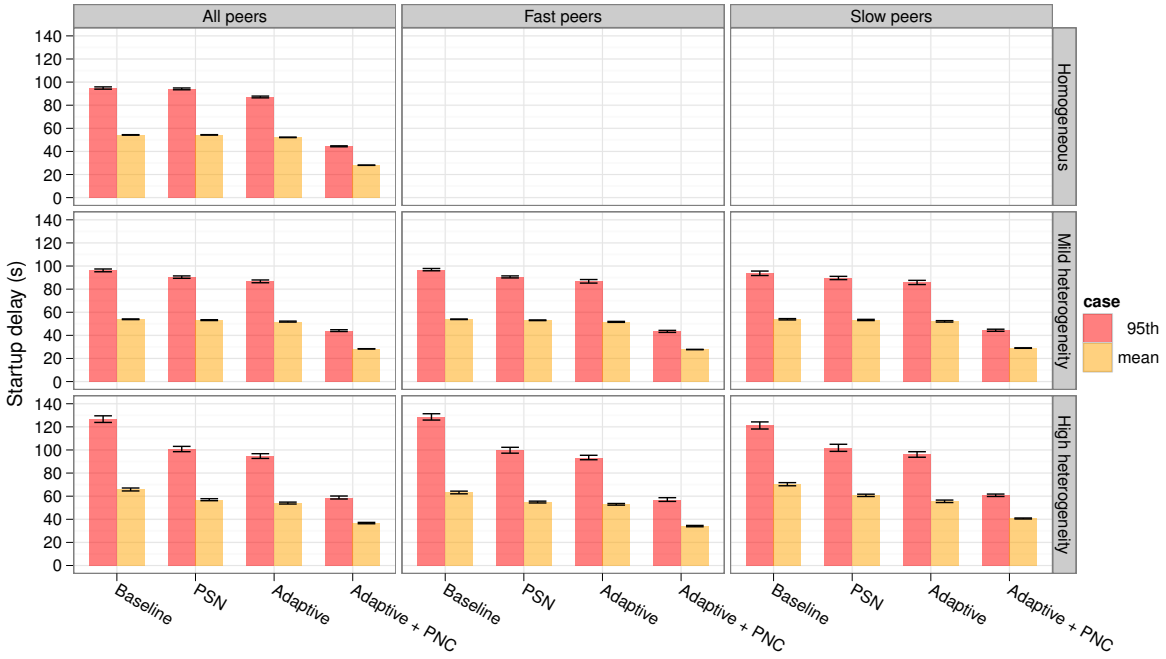

Fig. 3. Overprovision scenario: startup delay. The error bars correspond to a confidence level of 95%.

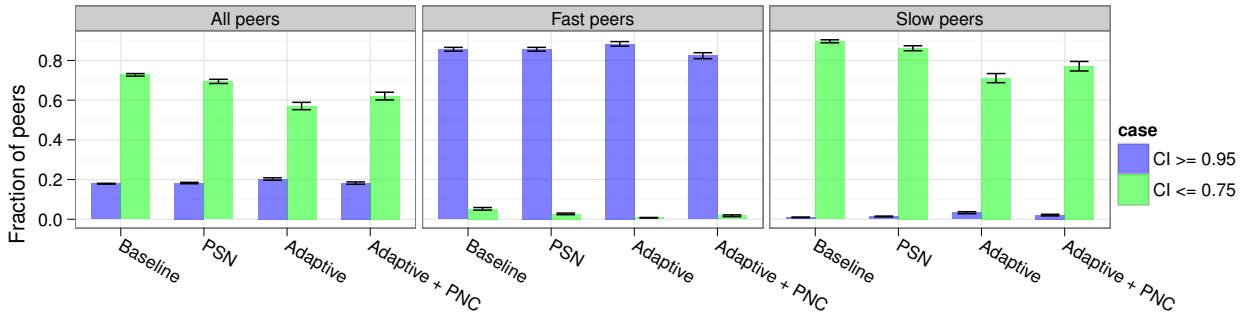

Fig. 4. Contention scenario: fraction of peers receiving good service ($CI \geq 0.95$) and poor service ($CI \leq 0.75$). The error bars correspond to a confidence level of 95%.

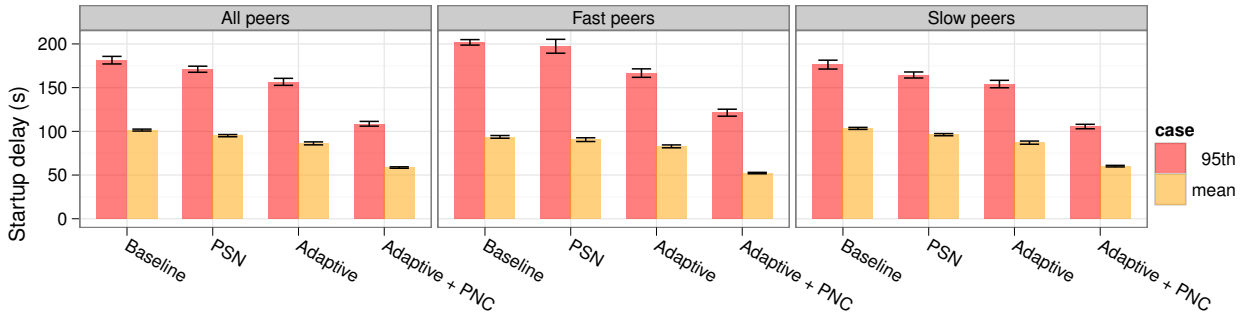

Fig. 5. Contention scenario: startup delay. The error bars correspond to a confidence level of 95%.

setup	μ_f/R	μ_s/R
Homogeneous	1.25	1.25
Mild heterogeneity	1.625	0.875
High heterogeneity	2	0.5

TABLE II
SETUPS FOR THE OVERPROVISION SCENARIO

- In the heterogeneous setups, using PSN or the adaptive policy already results in decreased startup delays for all peers. In fact, by opening more upload slots (PSN) or using them altruistically (adaptive), a higher fraction of peers will be chosen for optimistic unchoke, including newcomers. This phenomenon is more and more beneficial as the level of heterogeneity increases.

B. Contention scenario

In this scenario, we evaluate the ability of our strategies to face a situation where the average upload capacity is smaller than the video playback rate. We have again simulated a heterogeneous system with fast and slow peers. Similarly to the setup with high heterogeneity in the previous scenario, fast peers have upload capacity $\mu_f = 2R$ and slow peers have upload capacity $\mu_s = 0.5R$. However, this time, the percentage of fast peers is 20% and that of slow peers is 80%. This results in an average peer upload capacity of $0.8R$. The desired behavior in this case is that the fast peers will behave less altruistically in order to receive a good QoS themselves.

Figure 4 shows that, with all our strategies, fast peers are able to retain a good QoS. Slow peers, on the contrary, suffer of low QoS, although overall their service when using PSN and the adaptive policy is slightly better than when using the baseline G2G. It follows that, when bandwidth is scarce, there are incentives for cooperation. Furthermore, observing Figure 5, we can conclude that our proposed strategies are able to reduce the overall startup delays, despite the contention of resources in the system, with only little effect on the CI that peers obtain.

C. Dynamic scenario

In this scenario, we have tested the responsiveness of our adaptive policy to a sudden change in the total bandwidth capacity provided. More specifically, we simulated a system where peers have all initially an upload capacity of $2R$. Then, after 5000 seconds (ca. $3L$) from the start of the simulation, the capacity of 80% of these peers is suddenly reduced to $0.5R$. As a result, the average peer upload capacity changes from $2R$ to $0.8R$. This configuration, i.e. 80% slow peers and 20% fast peers, is then kept untill the end of the simulation. Figure 6 shows the average fraction of optimistic unchoke slots opened by fast peers in time. As we can observe, before the change in bandwidth supply, fast peers dedicate, on average, 0.4 of their slots to optimistic unchoke, which is the double of the minimum vaule, $\frac{1}{s_0} = 0.2$. After the change, we can observe that fast peers immediately start reducing the fraction of their optimistic unchoke slots, reaching a minimum of 0.2, in about 10 minutes. This value then gradually reaches 0.3.

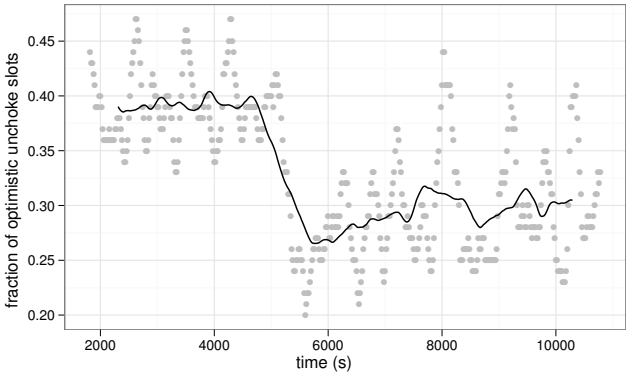

Fig. 6. Fraction of slots fast peers dedicate to optimistic unchoke in the dynamic scenario. A grey dot represents the average value across all peers at a given moment in time. The black line is a moving average, with window size 50, of these values.

VI. Conclusions and Future Work

In this paper, we have tackled the problem of improving QoS in BitTorrent-like VoD systems with heterogeneous peers through better peer selection policies. We have shown that, when the resources in the system are abundant in a heterogeneous system, rewarding peers proportionally to their upload capacity is neither (i) system-wide efficient, since less peers are granted a satisfactory QoS, nor (ii) individually profitable, since peers do not earn utility in downloading at rates much faster than the video playback rate. We have proposed a peer selection policy to overcome this effect, where nodes that are already receiving a high enough QoS adaptively increase the fraction of their bandwidth allocated to random peers. This strategy results in a more even distribution of bandwidth among peers and hence in a larger fraction of nodes receiving satisfactory QoS. At the same time, our strategy allows peers to become more selfish when resources are scarce, selecting less random peers. In this way, if there is resource contention, high-capacity nodes are still able to be prioritized and receive a good service, while low-capacity nodes or free-riders suffer of performance degradation. Furthermore, we have extended this strategy to decrease startup delays based on the idea of giving priority to newcomers. The average startup delay is reduced by up to 48% in our experiments, without negative impact on the stream continuity of other peers.

We note that, while there is no cost or performance degradation in using our peer selection policies, there are also no individual incentives for peers to adopt them. In situations where such incentives are necessary, our policies may be accompanied by long-term accounting (either centralized or decentralized [19]) of user behavior e.g. across different sessions. In this way, it is be possible to remember peers that have been altruistic in one session and reward them in the next session, for example by preferentially selecting them when they have just joined, and decreasing their startup delay.

Finally, although our policies have been evaluated using G2G, they are in principle applicable to any BitTorrent-based VoD system that uses the logic of choking and unchoking. We believe many such systems may benefit from the increase in the number of low-capacity peers that are able to receive QoS through the "help" of high-capacity peers in resource-abundant scenarios.

Acknowledgements

The research leading to this contribution has received funding from the European Community's Seventh Framework Programme in the P2P-Next project under grant no. 216217.

References

[1] The Tribler P2P System. http://www.tribler.org.
[2] A. Legout, G. Urvoy-Keller and P. Michiardi. Rarest First and Choke Algorithms are Enough. In *ACM IMC*, 2006.
[3] A. Legout, N. Liogkas, E. Kohler and L. Zhang. Clustering and Sharing Incentives in BitTorrent Systems. In *ACM SIGMETRICS*, 2007.
[4] A. Vlavianos, M. Iliofotou and M. Faloutsos. BiToS: Enhancing BitTorrent for Supporting Streaming Applications. In *IEEE Global Internet Symposium*, 2006.
[5] A.L. Jia, L. D'Acunto, M. Meulpolder, J.A. Pouwelse and D. Epema. Enhancing Reciprocity or Reducing Inequity: BitTorrent's Dilemma. In *IEEE CCNC*, 2011.
[6] S. Annapureddy, S. Guha, C. Gkantsidis, D. Gunawardena, and P. Rodriguez. Is high-quality vod feasible using P2P swarming? In *the 16th international conference on World Wide Web*, 2007.
[7] A.R. Bharambe, C. Herley and V.N. Padmanabhan. Analyzing and Improving a BitTorrent Networks Performance Mechanisms. In *IEEE INFOCOM*, April 2006.
[8] J. R. Douceur. The sybil attack. In *IPTPS*, 2002.
[9] A. Habib and J. Chuang. Service Differentiated Peer Selection: An Incentive Mechanism for Peer-to-Peer Media streaming. *IEEE Transactions on Multimedia*, 8(3), 2006.
[10] J.J.D. Mol. Free-riding Resilient Video Streaming in Peer-to-Peer Networks. *PhD Thesis, Delft University of Technology*, the Netherlands, January 2010.
[11] J.J.D. Mol, J. A. Pouwelse, M. Meulpolder, D.H.J. Epema and H.J. Sips. Give-to-Get: Free-riding-resilient Video-on-Demand in P2P Systems. In *SPIE MMCN*, 2008.
[12] M. Piatek, A. Krishnamurthy, A. Venkataramani, R. Yang, D. Zhang, A. Jaffe. Contracts: Practical Contribution Incentives for P2P Live Streaming. In *USENIX NSDI*, 2010.
[13] N. Carlsson and D. L. Eager. Peer-assisted on-demand Streaming of Stored Media using BitTorrent-like Protocols. In *IFIP NETWORKING*, 2007.
[14] N. Carlsson, D. L. Eager and A. Mahanti. Peer-assisted on-demand Video Streaming with Selfish Peers. In *IFIP NETWORKING*, 2009.
[15] M. Nowak. Five rules for the evolution of cooperation. In *Science 314, 1560*, 2006.
[16] P. Garbacki, D.H.J. Epema, J.A. Pouwelse and M. van Steen. Offloading Servers with Collaborative Video on Demand. In *IPTPS*, 2008.
[17] P. Savolainen, N. Raatikainen and S. Tarkoma. Windowing BitTorrent for Video-on-Demand: Not All is Lost with Tit-for-Tat. In *IEEE GLOBECOM*, 2007.
[18] P. Shah and J. F. Pris. Peer-to-Peer Multimedia Streaming using BitTorrent. In *IEEE IPCCC*, 2007.
[19] M. Piatek, T. Isdal, A. Krishnamurthy, and T. Anderson. One hop reputations for peer to peer file sharing workloads. In *USENIX NSDI*, 2008.
[20] J.A. Pouwelse, P. Garbacki, J. Wang, A. Bakker, J. Yang, A. Iosup, D.H.J. Epema, M. Reinders, M.R. van Steen, and H.J. Sips. Tribler: A social-based peer-to-peer system. *Concurrency and Computation: Practice and Experience*, 19, 2008.
[21] Y. Borghol, S. Ardon, N. Carlsson and A. Mahanti. Toward Efficient On-Demand Streaming with BitTorrent. In *IFIP NETWORKING*, 2010.
[22] Y. Yang, A.L.H. Chow, L. Golubchik and D. Bragg. Improving QoS in BitTorrent-like VoD Systems. In *IEEE INFOCOM*, 2010.
[23] Y.Huang, T.Z.J. Fu, D.H. Chiu, J.C.S. Lui and C. Huang. Challenges, design and analysis of a large-scale p2p-vod system. In *ACM SIGCOMM*, 2008.