

BitTorrent-like P2P Approaches for VoD: A Comparative Study

Lucia D'Acunto, Nitin Chiluka, Tamás Vinkó¹, Henk Sips

*Faculty of Electrical Engineering, Mathematics and Computer Science,
Delft University of Technology, Mekelweg 4, Delft 2628CD, The Netherlands*

Abstract

The enormous popularity of Video on Demand (VoD) has attracted substantial research attention into the effective use of peer-to-peer (P2P) architectures to provide solutions at large-scale. In particular, the high efficiency of BitTorrent has inspired many P2P protocols for VoD. However, these protocols use different approaches to adapt the design of Bittorrent to VoD, and in most cases their performance has been evaluated separately and in limited scenarios. As a consequence, the research community still lacks a clear understanding of how these protocols compare against each other and how well each of them would work in real world conditions, where, for instance, peers have heterogeneous bandwidths, may freeride or may be located behind NAT/firewall.

In this paper, we propose a simulation based methodology which aims at putting forward a common base for comparing the performance of these different protocols under a wide range of conditions. We show that, despite their considerable differences, (i) existing BitTorrent-like VoD approaches all share some characteristics, such as that their bandwidth reciprocity based methods to incentivize cooperation do not always yield an optimal overall performance. Furthermore, we demonstrate that (ii) in these protocols there is a trade-off between QoS and resilience to freeriding and malicious attacks. We also discover that, (iii) when peers doing streaming coexist with peers doing traditional file transfer, the latter actually benefit from this coexistence, at the expenses of the former. Finally, we show that (iv) early departures of peers from the system do not significantly affect the QoS de-

¹Permanent address: Institute of Informatics, University of Szeged, Hungary

livered, while jumping to a different position in the file has a bigger negative impact. Overall, our findings provide important implications for both VoD service providers and future system designers. On the one hand, our results can guide VoD service providers in selecting the most appropriate protocol for a given environment. On the other hand, exposing the flaws of current approaches will help researchers in improving them and/or designing better ones.

Keywords: Peer-to-Peer, Video-on-Demand, BitTorrent

1. Introduction

In recent years, significant research effort has focused on the effective use of a peer-to-peer (P2P) architecture to provide large-scale video-on-demand (VoD) services [18, 2, 8, 13, 41]. Current major VoD services like YouTube, Hulu and NetFlix, which supply hundreds of thousands of videos to millions of users everyday, make massive use of central servers for content provisioning. However, a P2P-based approach, with its natural scalability, could drastically cut the costs of VoD service providers. It has been demonstrated that, for instance, the MSN video server load can be reduced by roughly 95% through the use of a P2P-based approach [17]. Furthermore, non-profit organizations like Wikipedia², that cannot afford deploying the necessary servers to provide large-scale on-demand media services, would greatly benefit from such an approach [3].

Despite the potential benefits, providing on-demand services using a P2P approach is also a challenging task. In fact, similar to live P2P streaming systems, some quality-of-service (QoS) requirements have to be fulfilled, namely providing users with a high playback continuity and a short startup delay. However, the data access pattern in the two cases is different. In live streaming, peers have a shared temporal content focus, meaning that their playback positions are similar, while in the VoD case nodes might request videos at different times, and thus their playback positions would differ greatly.

Many of the early P2P systems were built to efficiently distribute large files among the participating users. Therefore, trying to adapt the designs of these systems to the streaming case came as a natural choice. In particular

²recently, Wikipedia has started a partnership with European project P2P-Next to provide videos and audio files on their pages with the help of a P2P infrastructure [27].

BitTorrent, which has been shown to make nearly optimal use of peers' upload bandwidth [4], has inspired many P2P protocols for VoD [8, 39, 33, 32, 25, 7]. In the context of traditional file transfer, BitTorrent achieves a high utilization of peers' upload bandwidth by means of a smart piece retrieval mechanism and strong incentives for cooperation³. The piece retrieval mechanism is based on a *local rarest-first* rule, where each peer prefers to download the pieces that are the rarest among its neighbors. Cooperation is incentivized by using a *direct bandwidth reciprocity* mechanism: peers select for uploading those peers that have uploaded to them at the highest rates in the past. However, BitTorrent was not designed for streaming and adapting its design to VoD poses two conflicting goals: (i) satisfying the fundamental QoS requirements for streaming, while (ii) still maintaining the high efficiency of the original BitTorrent protocol.

To date, a number of protocols has been proposed which tackle the problem by adapting to the VoD case the components of BitTorrent commonly acknowledged for its high efficiency: piece selection and peer selection. In particular, greater attention has been paid to piece selection, as the use of local rarest-first would result in long startup delays [28]. The piece selection policies for VoD proposed in literature can be broadly classified into: *window-based*, *probabilistic*, and *priority-based*. Window-based policies work by defining a sliding window, just ahead the playback position, within which pieces are downloaded, generally according to a local rarest-first rule. With probabilistic policies, pieces are downloaded according to some probability function, which normally is biased towards the first piece not yet downloaded. Finally, priority-based policies give priority to pieces which are close to being played. For what concerns peer selection, most VoD proposals [8, 39, 32, 7] maintain the default BitTorrent's policy, based on direct reciprocity. On the other hand, Mol *et al.* [25] argue that this policy is not the best fit for VoD applications, as it may be difficult for peers with lower level of progress reciprocate peers with higher level of progress. To remedy that, they propose a new peer selection policy based on *indirect reciprocity*, in which peers prefer uploading to other nodes that have forwarded pieces to others at the highest rate in the past.

³In contrast, the P2P protocols used by most commercial systems like PPStream and UUSee do not provide incentives for cooperation. Hence, users can just limit their upload contribution to the system or not contribute at all. This has caused these commercial systems to overprovision their network with a large number of servers [40].

However, all these approaches have been evaluated under different and limited scenarios. Hence, some methods might perform better than others under a certain set of conditions and worse under another. Furthermore, it is unclear how well each of them would work in real world conditions, where, for instance, peers have heterogeneous bandwidths and may reside behind a NAT or a firewall. Similarly, it is still unknown to what extent each approach really maintains the original BitTorrent’s incentives for cooperation and whether it is as secure against malicious attacks. Exposing the pros and cons of each approach can guide in selecting the most appropriate protocol to use in a given environment. Likewise, understanding the behavior of different BitTorrent-like VoD protocols will help researchers and system designers in improving current approaches and/or designing better ones.

In this paper, we take a first step towards answering these questions. We study and compare different peer selection and piece selection policies used by the BitTorrent-like VoD approaches proposed so far, under a wide range of conditions reflecting real world scenarios. Our analysis presents new insights into the impact of unconnectable and heterogeneous nodes, different malicious behaviors, coexistence with nodes doing traditional file-transfer and typical users’ watching behaviors.

Specifically, we make the following contributions:

1. We propose a simulation based methodology which aims at putting forward a common base for comparing the performance of different BitTorrent-like P2P protocols for VoD under a wide range of conditions (Section 5);
2. We find out that, in general, a trade-off exists between QoS and freeriding resilience and between QoS and security (i.e. a more QoS oriented design is more susceptible to freeriding and/or malicious attacks); moreover, we identify a number of ways in which malicious peers can undermine the performance of the system (Sections 6-7);
3. We discover that the current approaches to incentivize cooperation result in overall bad performance when peers have heterogeneous bandwidths or are unconnectable (i.e. behind a firewall or NAT) (Sections 8-9);
4. We show that the coexistence of nodes doing streaming and nodes doing traditional file transfer is disadvantageous for the former and advantageous for the latter (Section 10);
5. Furthermore, we consider the impact of typical watching behaviors of

- P2PVoD systems' users and we show that peers departing early do not influence much the delivered QoS, while peers jumping to a different position in the file can have a bad influence for the probabilistic and window-based policies (Section 11);
6. Finally, we discuss the implications of our findings for future system designs and for VoD service providers (Section 12).

2. Related Work and Motivation

Previous works on P2P VoD systems mostly focused on studying and improving their performance, under the assumption that every peer would donate all its upload capacity and could communicate with every other peer in the system. Early studies [17, 28, 14], for example, show that, under this assumption on the nodes, the P2P approach is potentially effective in distributing on-demand content and providing users with good QoS. On the same line of work, Yang *et al.* [41] focus on the load balancing problem among P2P nodes as well as on the efficient scheduling of piece requests in order to further enhance QoS.

However, we note that these previous efforts do not take into account some important factors that can drastically reduce the performance and scalability of P2P systems. The first of these factors is user contribution. It has been argued that peers will not contribute their resources (namely files and upload bandwidth) unless they are given an incentive to do so. A study conducted on the Gnutella P2P system [1] seems to confirm this hypothesis, as it reports that a high percentage ($> 70\%$) of users freeride (at the time, Gnutella did not provide any incentives for users to contribute). On the other hand, it has been shown that in BitTorrent, which has an embedded incentive mechanism, only 10% of the users freeride [43]. BitTorrent also makes nearly optimal use of peers' upload bandwidth [4]. Therefore, it is no surprise that it has attracted a lot of research in the past decade, and many recent works on P2P VoD have been inspired by its design [8, 39, 33, 32, 25, 7]. Closely related to users' willingness to contribute is the problem of malicious attacks, which has also been mostly ignored in previous research on P2P VoD. However, systems that are not robust against attacks might suffer from overall performance degradation and fail in providing good QoS to their users.

Another important aspect that received little attention in P2P streaming literature is the heterogeneity of P2P nodes. Part of this heterogeneity is

intrinsically related to users' Internet access means, which may result in different bandwidth capacities as well as reduced connectability of some nodes (hosts residing behind a firewall or a NAT, for example, cannot receive inbound connections, unless the firewall/NAT is configured to do so). In open systems there is also another kind of heterogeneity related to the protocol used. In the BitTorrent ecosystem, for example, it is not uncommon to have some nodes performing traditional file transfer, while some others are requesting the same file in streaming mode (nowadays, the streaming functionality is supported by many BitTorrent clients, such as BitTorrentDNA [6], μ Torrent [37], and Tribler [36]).

To the best of our knowledge, there is only one previous work that recognized the importance of some of the aforementioned aspects for VoD systems [34]. In particular, the authors focus on incentives and peer unconnectability. To incentivize users to contribute, they propose the setup of a special infrastructure to keep track of individual peer contribution. For what concerns unconnectable nodes, they introduce an approach to make these peers discoverable by connectable ones. However, we will show in Section 9 that, already when the fraction of unconnectable nodes is a mere 30%, this is not enough to solve the problem.

The study presented here is different and complementary to these previous efforts in that it aims at understanding to what extent a P2P approach is suitable for VoD, when the aforementioned factors come into play. In particular, we consider a BitTorrent-based approach, due to BitTorrent's characteristics of having built-in incentives for user contribution and efficient peers' upload bandwidth usage.

3. Background on 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. Corresponding to each file available for download, there is a central component called tracker that keeps track of the nodes currently in the system. When a new peer joins, it contacts the tracker to obtain a list of a random subset of these nodes. Some implementations of the BitTorrent protocol also make use of a DHT (which was not part of the original design), next to the tracker, to enhance peer discovery.

Each node then 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 equally divides its upload capacity into a number of *upload slots*. There are two types of upload slots: *regular unchoke slots* and *optimistic unchoke slots*. Regular unchoke slots are assigned according to a strategy based on direct reciprocity: peers prefer other nodes that have recently provided data to them 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.

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 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 [20].

4. Protocol Policies for BitTorrent-like VoD

In this section, we give an overview of the piece selection and peer selection policies used in the BitTorrent-like VoD protocols we consider.

4.1. Piece Selection Policies

A piece selection policy determines the next piece of the video file a peer selects for download. The piece selection policies for VoD proposed in literature try to find a trade-off between in-order download (necessary for QoS) and high bartering opportunities among peers (to ensure an efficient peer bandwidth utilization). In order to do so, they are all equipped with a *sequentiality parameter*, which can be tuned prior to deployment to favor one aspect or the other. In this work, we consider three categories of piece selection policies: 1) window-based 2) probabilistic, and 3) priority-based, as described below.

4.1.1. Window-based piece selection (WIN)

Window-based solutions typically employ a sliding window within which pieces are chosen [33]. The window advances from the beginning to the end of the file according to the sequential download progress at the local peer. Normally, the window starts at the first piece not yet downloaded. Within the window, rarest-first piece selection is often applied. Naturally, a smaller window ensures close to sequential piece retrieval, but reduces piece diversity and, hence, bartering opportunities among peers. On the other hand, a larger window allows for higher bartering opportunities among peers but increases users's startup delays and their chance of downloading pieces before their playback is due. The size (in pieces) of the window represents the sequentiality parameter of this piece selection policy and we denote it with w .

4.1.2. Probabilistic piece selection (PROB)

In probabilistic piece selection, pieces are chosen in relation to some probability distribution, generally with a bias towards the first pieces not yet downloaded. In this work, we consider the policy proposed in [9], where a Zipf probability distribution is used. Specifically, the probability that a peer a selects to download a piece k is proportional to $(k + 1 - k_0)^{-\theta}$ where k_0 is the index of the first piece peer a has not yet downloaded. Similar to the window size w for the window-based policy, θ represents the sequentiality parameter of this piece selection policy and can be tuned to provide close to sequential piece retrieval, but low bartering opportunities among peers (large θ), and vice versa (small θ).

4.1.3. Priority-based piece selection (PRIO)

With priority-based approaches, priority is given to pieces which are close to be played. We use the method presented in [25], where a peer, whose current playback position is p , will request a piece i on the first match in the following list of sets of pieces (known as *priority sets*):

- high priority: $p \leq i < p + h$: in-order piece selection if the local peer has already started playback, rarest first otherwise;
- mid priority: $p + h \leq i < p + 5h$: with rarest first piece selection;
- low priority: $p + 5h \leq i$: with rarest first piece selection.

In these definitions, h denotes the size (in pieces) of the high priority set and represents the sequentiality parameter of this piece selection policy. Similarly to the previous policies, the parameter h can be tuned to give more emphasis to sequential piece retrieval (large h) or high bartering opportunities among peers (small h).

4.2. Peer Selection Policies

A peer selection policy determines how a node selects another node to upload data to. In BitTorrent-like systems, the peer selection policy has the task of incentivizing peer cooperation, and therefore it is usually designed to favor good uploaders. In this work, we will consider two policies, based on *direct reciprocity* and *indirect reciprocity*, respectively.

4.2.1. Direct Reciprocity

When a node uses peer selection based on direct reciprocity, it will upload to other nodes that have recently uploaded to it at the highest rates. As mentioned in Section 3, this is the standard peer selection policy employed in BitTorrent.

Direct reciprocity is easy to implement: each peer takes its decisions based only on the information locally available (i.e. the measured upload rates of other nodes) and no long-term memory is required (normally peers re-evaluate the upload rates of other nodes every 10 seconds). Many studies show that it works successfully in the context of traditional file transfer [43, 19, 20]. However, it has been argued that in P2P VoD systems, due to the somewhat in-order download progress of peers, it is more difficult for peers with lower degrees of progress to reciprocate peers with higher degrees of progress [25]. Consequently, it would be more difficult for a peer to detect whether another peer is freeriding or simply has no interesting piece to barter for.

4.2.2. Indirect Reciprocity

When a node uses peer selection based on indirect reciprocity, it will upload to other nodes that have recently *forwarded* data to others at the highest rates. In this work, we use the method introduced in the *give-to-get* (G2G) protocol [25]. In G2G, a peer a discovers the forwarding rate of a child node b by periodically asking its grandchildren about the pieces received from b . Note that the child b is not asked directly as it could make false claims. This process is depicted in Figure 1.

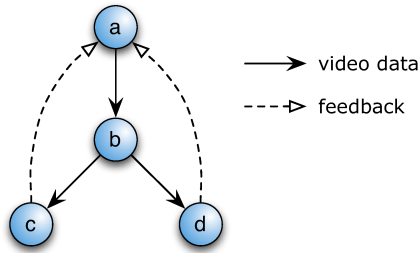


Figure 1: Data flow and feedback flow for indirect reciprocity.

Indirect reciprocity is intuitively more suitable to a VoD scenario, since it does not necessarily require nodes with lower levels of progress to reciprocate nodes with higher levels of progress. However, since each peer needs to gather information from other nodes, it is more costly to implement (especially in presence of NATs and firewalls) and is potentially more vulnerable to various types of attacks, as we will show in Section 7.

5. Methodology

In this section, we describe the approach we use to compare and study different VoD protocols. First, we introduce some of the terms we will employ in our analysis, as well as the model of the system we consider. Then, we illustrate in detail the experimental setup for our analysis and how we tuned the sequentiality parameters of the piece selection policy considered.

5.1. Definitions

In this subsection we introduce the most important definitions which are used throughout the whole paper. The first three definitions are related to user behavior, which are followed by definitions closer to the system level.

A peer whose upload capacity is set to 0 is called a *freerider*. A freerider follows the specific protocol in all aspects. Freeriders generally correspond to users who configure their network access in order not to share their upload capacity. The behavior of freeriders is termed as *freeriding*. A *malicious peer* does not follow (some of) the rules of the specific protocol. Finally, an *honest peer* follows all the rules of the specific protocol and has upload capacity larger than 0. Honest peers do not try to manipulate the protocol and, whenever possible, share all their upload capacity.

An *unconnectable peer* does not possess a globally reachable address. Usually, this happens when the peer resides behind a NAT or a Firewall which is not configured to accept incoming connections. A *missed piece* is a piece whose download cannot be completed before the time it is due to be played. A peer a is said to be *interested* in another peer b when a possesses a piece that b does not and has not missed. Similarly, a is said to be *interesting* for b .

5.2. System Model

We consider a system where the participating peers can retrieve the stream of a particular video file, of playback rate R and size F , which is split into n pieces of identical size. Each peer maintains on disk a copy of all the pieces it has ever downloaded, so that it can serve them to others. The system is assumed to be in steady state⁴, with peers joining at a constant rate λ . Peers have an average upload capacity denoted by μ and, for the purpose of the analysis, we assume that their download capacity is not a bottleneck and can be considered to be infinite. In addition to the peers, the system contains a number of servers (or *seeders*) which contribute an aggregate (constant) upload capacity of U_s . Based on the watching behavior observed in recent measurement studies of P2PVoD systems [22], we assume that users play the video sequentially from the beginning. Furthermore, we assume that peers leave as soon as their download is complete and, at any point in time, there is no other peer seeding the content of the video file besides the servers. Although this scenario may not be realistic in practice, minimizing the bandwidth supplied to the peers will make the distinction between the performance of various policies more visible.

Given the above notation, it is easy to see that the expected download speed u of a peer in steady state is

$$u = \frac{U_s}{N} + \mu, \quad (1)$$

where N is the number of peers in the system. From Little's Law it follows that

$$N = \lambda m, \quad (2)$$

⁴a system is said to be in *steady state* when, although peers might join and leave, the total number of peers remains constant over time.

where m is the average residence time of peers in the system.

Based on this observation, it is clear that, if the server bandwidth U_s is constant, the average download speed u of peers in a system with a low workload (i.e. low arrival rate or short residence time) is larger than in a system with a high workload (high arrival rate or long residence time). This would lead to an unfair comparison among scenarios characterized by different workloads, e.g. peers in scenarios with low arrival rates would achieve faster download speeds and, consequently, better QoS, than peers in scenarios with higher arrival rates. To make the comparison fair, the server bandwidth U_s needs to be dimensioned to the system workload such that, in all scenarios, peers reach a certain desired steady state download speed denoted with u^* and therefore

$$U_s = (u^* - \mu) \lambda m,$$

where this formula is obtained by combining Eqs. (1) and (2). For the system to be able to provide a good QoS, it is necessary to have $u^* \geq R$. In particular, the closer u^* is to R , the lower the server bandwidth. If we express u^* as a function of R as follows

$$u^* = \gamma R, \text{ with } \gamma \geq 1$$

then the required server bandwidth U_s can be calculated as

$$U_s = (\gamma R - \mu) \lambda m. \quad (3)$$

In the particular case where peers leave as soon as their download is complete, their average residence time in the system, m , equals to the expected download time of the entire video file, i.e. $m = F/u^*$. Then Eq. (3) can be rewritten as

$$U_s = \left(1 - \frac{\mu}{\gamma R}\right) \lambda F. \quad (4)$$

In this way, once the characteristics of the system are known, the VoD service provider only needs to set a value for the γ parameter that suits the needs of the system.

5.3. Experimental Setup

We compare the performance of different schemes of peer and piece selection policies by means of simulations within a wide range of aspects and scenarios representative of the real world. Our simulation approach is described below.

5.3.1. *Simulation Environment*

For this purpose, we have extended the BitTorrent simulator designed by Microsoft Research [4], in order to support VoD. This is a very detailed simulator, where all the elements of a BitTorrent system are modelled with great accuracy, from the creation of the overlay to the exchange of piece between peers. The overlay is created by means of a tracker module operating in the classical BitTorrent fashion outlined in Section 3: when it is contacted, the tracker returns a list of random nodes to the requesting peer. Although the end-to-end delay is not modelled in this simulator, we believe that this simplification does not have significant impact on our results due to following reasons. First, the end-to-end delay is usually an order of magnitude or two smaller than the time required to download a piece, the basic unit of data transfer. A measurement study [44] shows that over 90% of all pairs of PlanetLab and King dataset [46] hosts have round-trip latency of less than 200 ms and 300 ms respectively. Another study [45] shows that nearly 80% of all pairs of Gnutella peers have round-trip latency of less than 280 ms. In comparison, the time required by one peer to download a piece of 256 KiB from another peer whose upload rate to the former is 200kbps (which is the typical scenario in our paper) is over 10 seconds. Second, BitTorrent’s pipelining mechanism [10] masks much of the control traffic latency in practice. Furthermore, the simulator does not model packet-level dynamics of TCP connections but rather it assumes that the connections traversing a link share the link bandwidth equally, with the share of each connection fluctuating as the number of connections varies. Although this simplification means that “TCP anomalies” (e.g. timeouts) are not modeled, the occurrence of these anomalies would impact all the policies considered in an equal measure and, hence, would not affect the relations between them (i.e. if a policy performs better than another when we do not consider TCP anomalies, it will still be so when we consider them). Due to its great level of accuracy in reproducing the behavior of BitTorrent systems, this simulator has been widely used, also for simulating BitTorrent-like VoD protocols [7, 11, 41]. Next to all the elements of the original BitTorrent protocol, our extension also supports all the piece selection policies presented in Section 4.1 and allows for the system to either adopt direct reciprocity or indirect reciprocity as peer selection policies. We have made our extension available at <http://www.pds.ewi.tudelft.nl/dacunto/research> for those interested in continuing this research.

The settings for our simulations are shown in Table 1. In this setup, we assume that the peer upload capacity μ equals to $1000kb/s$ and consequently we set the video playback rate R to a value just below that, $800kb/s$, in order to avoid that peers experience a bad playback continuity due to bandwidth contention. This video playback rate is within the typical range of P2P VoD systems [16]. In general, to guarantee a high QoS, VoD service providers should stream videos at a playback rate that can be sustained by the average peer upload bandwidth. For the video duration, we choose $50min$, as this represents a common value for the video streamed in commercial systems [18, 22, 42]. For what concerns the piece size, usually a smaller piece size determines a faster propagation of pieces among peers in the system. However, as noted in [32], a small piece size also increases the communication overhead (because more piece requests and announces of piece received have to be sent out). Many BitTorrent clients (e.g. Azureus and Transmission) recommend a piece size of $250kB$ for files of size in the range of $300-700MB$, and therefore we adopt this value here. For the computation of the server bandwidth U_s , we use $\gamma = 1.3$, in order to compensate for fluctuations in download speeds and to account for the fact that downloaders are not always able to upload at their full capacities. To decide when playback can safely commence, we use the strategy introduced in [8]. Specifically, a peer will start playback only when it has obtained all the pieces in an initial buffer of size B and its current *sequential progress*⁵ is such that, if maintained, the download of the file will be completed before playback ends. The buffer size B is equal to w or h for the window-based and the priority-based piece selection policies, respectively, and is set to 20 pieces for the probabilistic one.

5.3.2. Aspects affecting the performance of P2PVoD systems

In this work, we consider the influence of six main aspects, typical of today's internet and user behavior as pictured in Figure 2, on the performance of a P2PVoD systems:

1. freeriding
2. malicious attacks

⁵a peer's *sequential progress* is the speed at which the index of the first piece in the file not yet downloaded grows and it represents the rate at which a continuous stream is received [28]. It should not be confused with the sequentiality parameter (defined in Section 4.1) that characterizes each piece selection policy.

Table 1: Simulation Settings

Parameter	Value
Video playback rate R	800 kb/s
Video length L	50 min
Simulation time	between 250 and 750 min ($5L$ and $15L$)
Piece size P_S	256 kB
Initial buffer B	20 pieces (PROB) / w (WIN) / h (PRIO)
Upload rate μ	1000 kb/s ($1.25R$)

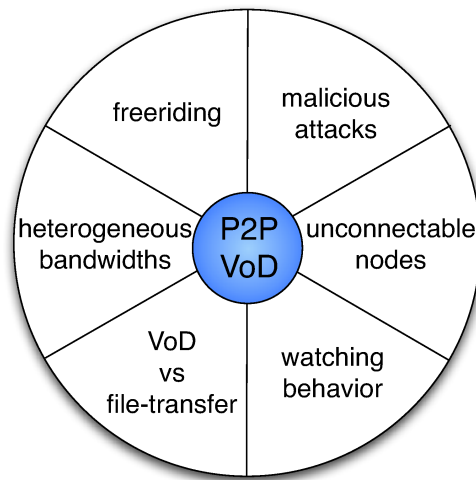


Figure 2: Aspects affecting the performance of P2PVoD systems.

3. heterogeneous peer bandwidths
4. presence of unconnectable nodes
5. coexistence of peers doing VoD with peers doing traditional file-transfer
6. watching behavior

As we have observed in Section 2, the lack of user contribution is one of the crucial aspects that can affect the performance of P2PVoD systems. In this work, we consider two ways in which users can avoid to contribute: by means of freeriding and by means of malicious attacks. Furthermore, we account for the heterogeneity of peer bandwidths, and the fact that some nodes reside behind NATs or firewalls. In addition, we examine how the coexistence of peers doing VoD with peers doing traditional file-transfer affects the

performance of either group of peers. Finally, we also explore some watching patterns typical of the users of P2PVoD systems, such as jumping to a different part of the video or leave early, and evaluate their impact. In order to be able to distinguish the influence of a specific aspect on the performance of P2PVoD systems, we analyze the impact of each of them separately.

5.3.3. Performance Metrics

We analyze the performance of any combination of peer and piece selection policies using the following two metrics:

- *continuity index*, defined as the ratio of pieces received before their deadline over the total number of pieces;
- *startup delay*, defined as the time a user has to wait before playback starts.

Each simulation run is executed 15 times, and then average values and confidence intervals (with confidence level of 95%) for the above metrics are computed. The lower the arrival rate, the longer it takes for the system to reach the steady state. Therefore, simulations times are longer for lower arrival rates. Furthermore, for the results we have only considered the peers who have joined after the first half of the simulation time and before the last L minutes from the simulation end.

5.4. Tuning the Sequentiality Parameters for the Comparison

Recall from Section 4.1 that each piece selection policy for VoD is characterized by a sequentiality parameter which allows one to give more emphasis to sequential piece retrieval or to high bartering ability among peers. In order to fairly compare these policies against each other, they need to be “tuned” in such a way that they exhibit a similar level of sequentiality. To do so, we have performed several experiments where each policy uses a wide range of values for its sequentiality parameter. For these experiments, the fraction of freeriders was set to 10%, as this represents a typical value in BitTorrent systems [43] and peer arrival rates are as follows: $\boldsymbol{\lambda} = (\lambda_0, \lambda_1, \lambda_2, \lambda_3) = (0.005, 0.01, 0.05, 0.1)$ peers/s. The smaller values (0.005, 0.01) account for the case of not very popular videos, which generate only little load in the system, while the larger values account for the case of very popular video, which thus generate higher load.

To find a common baseline, we use the following approach: we select the sequentiality parameter for each piece selection policy such that the startup delays experienced by honest peers within each of them are similar. We first focus on the direct reciprocity case and then, based on it, we tune the sequentiality parameters for the case of indirect reciprocity.

5.4.1. Direct Reciprocity

Given the setup introduced in Section 5.3, we have tested the following values for the sequentiality parameters:

- w : from 10 to 60, with step 5 (this set is denoted by W);
- θ : from 1.5 to 4, with step 0.25 (this set is denoted by Θ);
- h : 10, to 60, with step 5 (this set is denoted by H).

As we can observe from Figure 3, the instances that produce short startup delays generally determine low continuity index as well. This confirms that a trade-off indeed exists between sequentiality and bartering ability among peers (and thus between short startup delay and high continuity index).

To select our policy instances, we start from the window-based one. For this policy, increasing the sequentiality parameter w means longer startup delays but also higher continuity index (Figure 3 and Section 4.1). We select the smallest sequentiality parameter w for which at most 10% of the honest peers are experiencing a continuity index less than 0.95. This choice is motivated by Habib *et al.* [15]: the user’s overall perceived video quality is considered very good when the continuity index of a stream is not less than 0.95. This selection leads to $w^* = 40$.

Then, we calculate the Euclidean distance between the startup delay values of this particular instance of the window-based policy and the startup delay values of the other two policies. More specifically, having denoted with $\mathbf{D}^{w^*} = (D_0^{w^*}, D_1^{w^*}, D_2^{w^*}, D_3^{w^*})$ the vector containing the values for the startup delays obtained with arrival rates λ for the window-based policy characterized by $w^* = 40$, and with $\mathbf{D}^\theta = (D_0^\theta, D_1^\theta, D_2^\theta, D_3^\theta)$ and $\mathbf{D}^h = (D_0^h, D_1^h, D_2^h, D_3^h)$ the startup delay vectors for the generic probabilistic and priority-based policies, respectively, we find the vectors \mathbf{D}^{θ^*} and \mathbf{D}^{h^*} most similar to \mathbf{D}^{w^*} as follows:

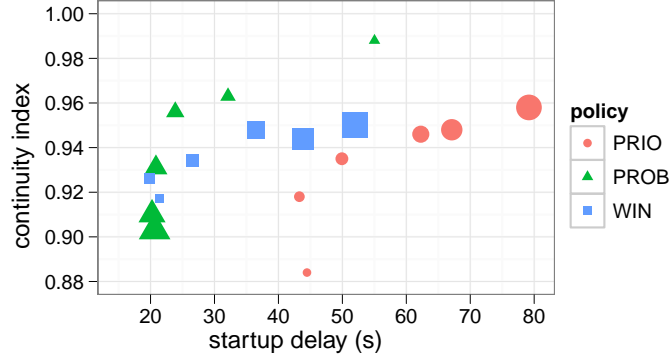


Figure 3: A scatter plot of startup delay vs continuity index for different piece selection policy instances. Smaller points correspond to smaller values for the sequentiality parameters of each piece selection policy. Peer arrival rate is 0.005 peer/s and the remaining simulation settings are as in Table 1.

$$\mathbf{D}^{\theta^*} = \min_{\theta \in \Theta} \sqrt{\sum_{k=0}^3 (D_k^{w^*} - D_k^\theta)^2}, \quad \text{and} \quad \mathbf{D}^{h^*} = \min_{h \in H} \sqrt{\sum_{k=0}^3 (D_k^{w^*} - D_k^h)^2}.$$

Note that technically this means that for all parameters from the sets Θ and H we take the magnitude of the four-dimensional vectors \mathbf{D}^θ and \mathbf{D}^h and choose the one which has the most similar magnitude to that of \mathbf{D}^{w^*} . Using this method, we obtain that the window-based policy characterized by $w^* = 40$ is mostly similar to the probabilistic and the priority-based policies characterized by $\theta^* = 2$ and $h^* = 25$, respectively. The Euclidian distances for these and other values of the sequentiality parameters are shown in Table 2. Figure 4 plots the startup delay and the continuity index for honest peers when the three policies use these selected values for their sequential parameters. Because we have used a discrete set of values for the sequentiality parameters, we would like to evaluate how close to the "optimal" our solutions are. To do so, we have calculated, for both priority based and probabilistic policies, the euclidian distance between the selected vectors and those obtained by using the next higher and next lower value of the sequentiality parameters. These results are reported in Table 3. As we can observe, the distances of both \mathbf{D}^{θ^*} and \mathbf{D}^{h^*} from their predecessors and successors

Table 2: Euclidian distances to D^{w^*} for different sequentiality parameters (direct reciprocity)

Priority-based		Probabilistic	
$h^* = 20$	12s	$\theta^* = 1.75$	45s
$h^* = \mathbf{25}$	8s	$\theta^* = \mathbf{2}$	15s
$h^* = 30$	16s	$\theta^* = 2.25$	43s

Table 3: Euclidian distances to D^{h^*} and D^{θ^*} for different sequentiality parameters (direct reciprocity)

Priority-based		Probabilistic	
$h^* = 20$	10s	$\theta^* = 1.75$	44s
$h^* = 30$	12s	$\theta^* = 2.25$	40s

are quite similar, hence giving an idea that the values we found are very close to the optimal.

5.4.2. Indirect Reciprocity

In the case of indirect reciprocity we have chosen, for each piece selection policy, a value for the sequentiality parameter such that the startup delay of honest peers is similar to that of the same policy in the case of direct reciprocity. To evaluate similarity, we have used again the Euclidean distance. This approach led to the following values: $w^* = 30$, $\theta^* = 2$ and $h^* = 20$, which, as we can see in Figure 4, exhibit equal or slightly lower startup delays than their respective in the case of direct reciprocity.

The selected values for the sequentiality parameters for both indirect and direct reciprocity are summarized in Table 4. Unless otherwise stated, we will use these values in the remainder of this paper.

Table 4: Setups for the Sequentiality Parameters

	Direct reciprocity	Indirect reciprocity
h^*	25	20
θ^*	2	2
w^*	40	30

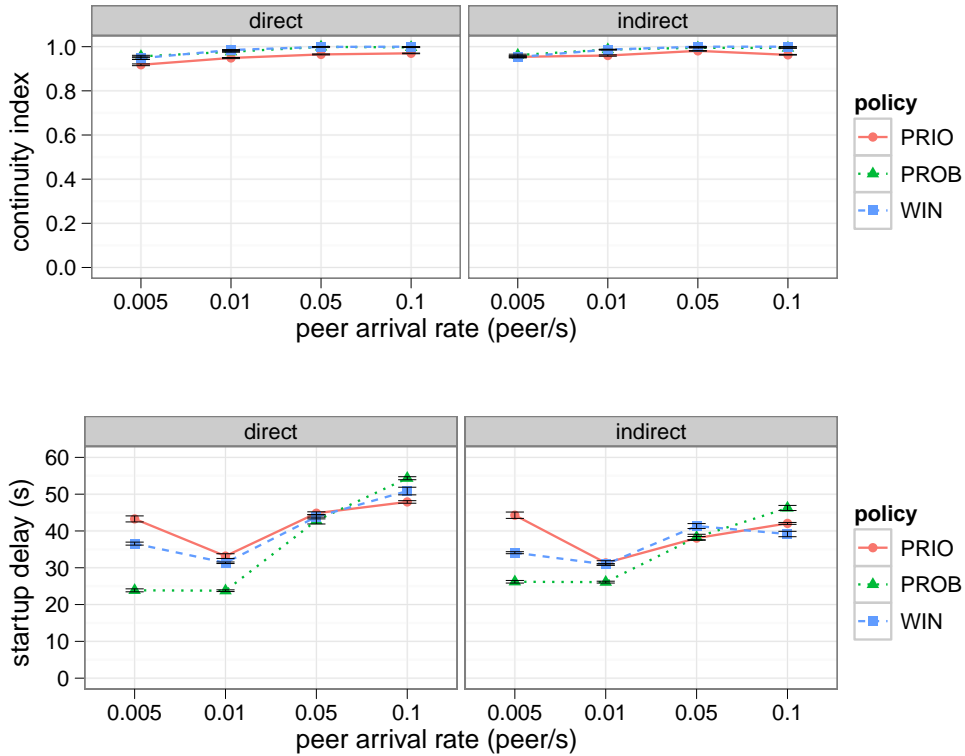


Figure 4: Performance of honest peers with piece selection policies using the selected sequentiality parameters (Table 4). The simulation settings are as in Table 1 and freeriders are 10% of the population.

6. Freeriding Resilience

One of the key factors for the success of BitTorrent is its effective incentive mechanism which induces peers to contribute bandwidth while downloading. This has motivated researchers to apply the same design to the VoD case. However, to the best of our knowledge, the research community still lacks of a clear understanding of whether, once applied to VoD, this mechanism yields to the same degree of freeriding resilience as the original BitTorrent protocol. In order to verify that, in this section, we analyze the performance of each combination of piece and peer selection policy as we vary the fractions of freeriders. We will first analyze the case of direct reciprocity and then the case of indirect reciprocity.

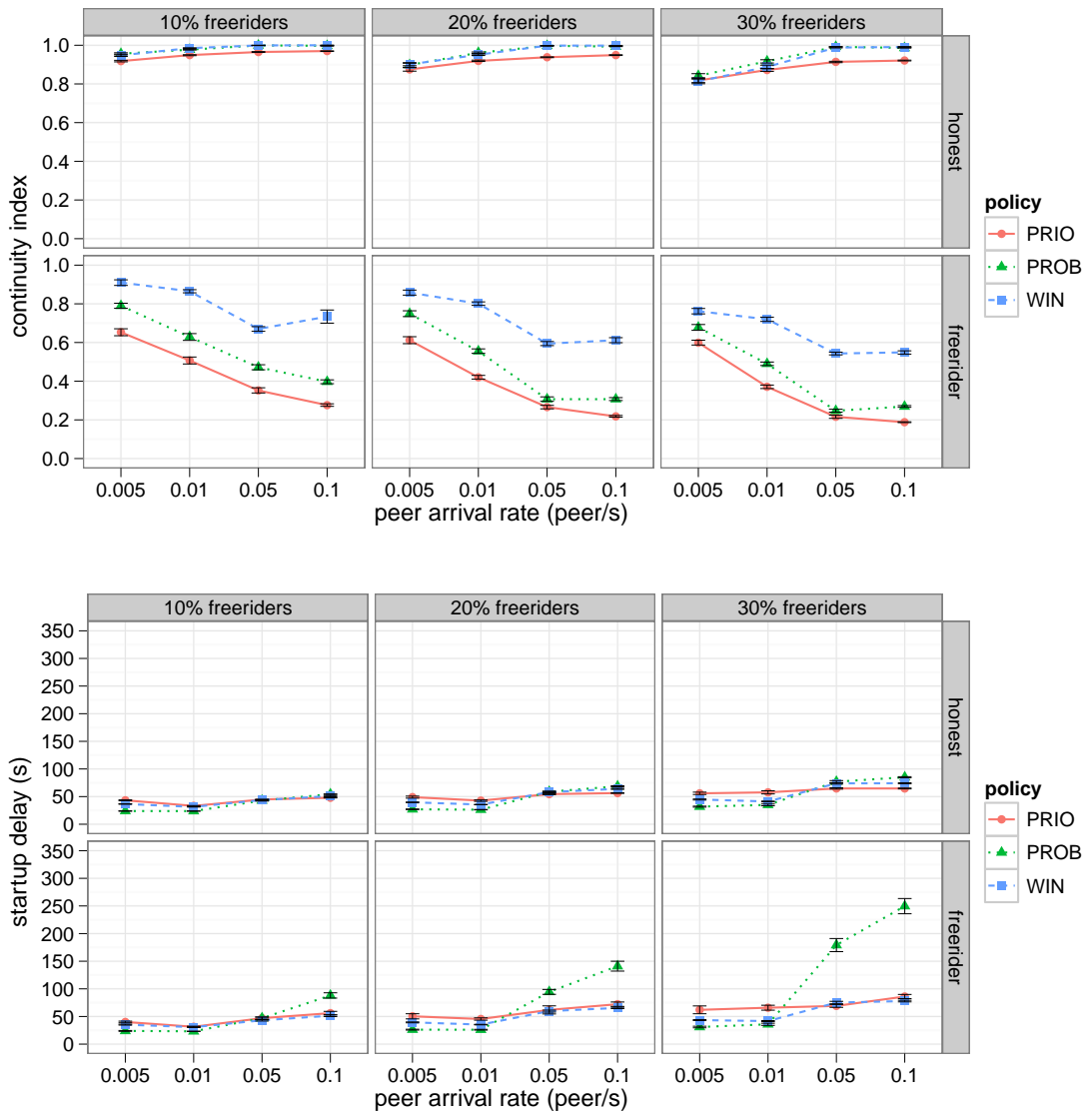


Figure 5: Performance of honest and freeriding peers in a system using direct reciprocity with different fractions of freeriders.

6.1. Direct reciprocity

Figure 5 plots the continuity index and startup delay for different fractions of freeriders. As we can observe, when the fraction of freeriders is small (10%), all the policies succeed in providing optimal playback continuity (con-

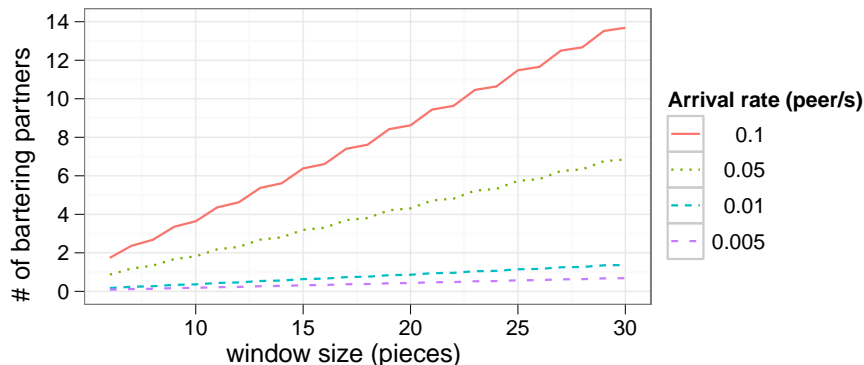


Figure 6: Expected number of bartering partners for the window-based piece selection policy with different window size and peer arrival rate. Parameters are from Table 1.

tinuity index almost 1) to honest peers. However, as the fraction of freeriders increases, their continuity index worsens, especially for lower arrival rates. This can be explained as follows. When the arrival rate is low, newly joined peers have a harder time in reciprocating older peers, because the latter have a significantly higher level of progress. As a consequence, it is difficult for an older peer p to distinguish between a freerider and a newly joined peer, who, although willing to reciprocate, is not doing so because it does not have any interesting piece for p . Intuitively, reducing the sequentiality of the piece selection policy will increase the bartering ability of peers, thus improving freeriding resilience. We have analyzed in more detail the relationship between sequentiality and number of bartering partners for different peer arrival rates by means of mathematical analysis, for which we refer the reader to the Appendix. Figure 6 demonstrates our analysis for the window-based piece selection policy; with low arrival rate even high window size would result in only a few bartering partners.

For what concerns the specific piece selection policies, the probabilistic one performs the best, providing the highest continuity index for honest peers, and relatively low continuity index for freeriders. The priority-based policy, on the other hand, determines the worst continuity indexes for honest peers. This is due to the fact that, different from the window-based and probabilistic ones, in the priority-based policy, the download point is relative to the current playback position, rather than the first piece not yet downloaded. Hence, there is a “less safe” distance between the download progress

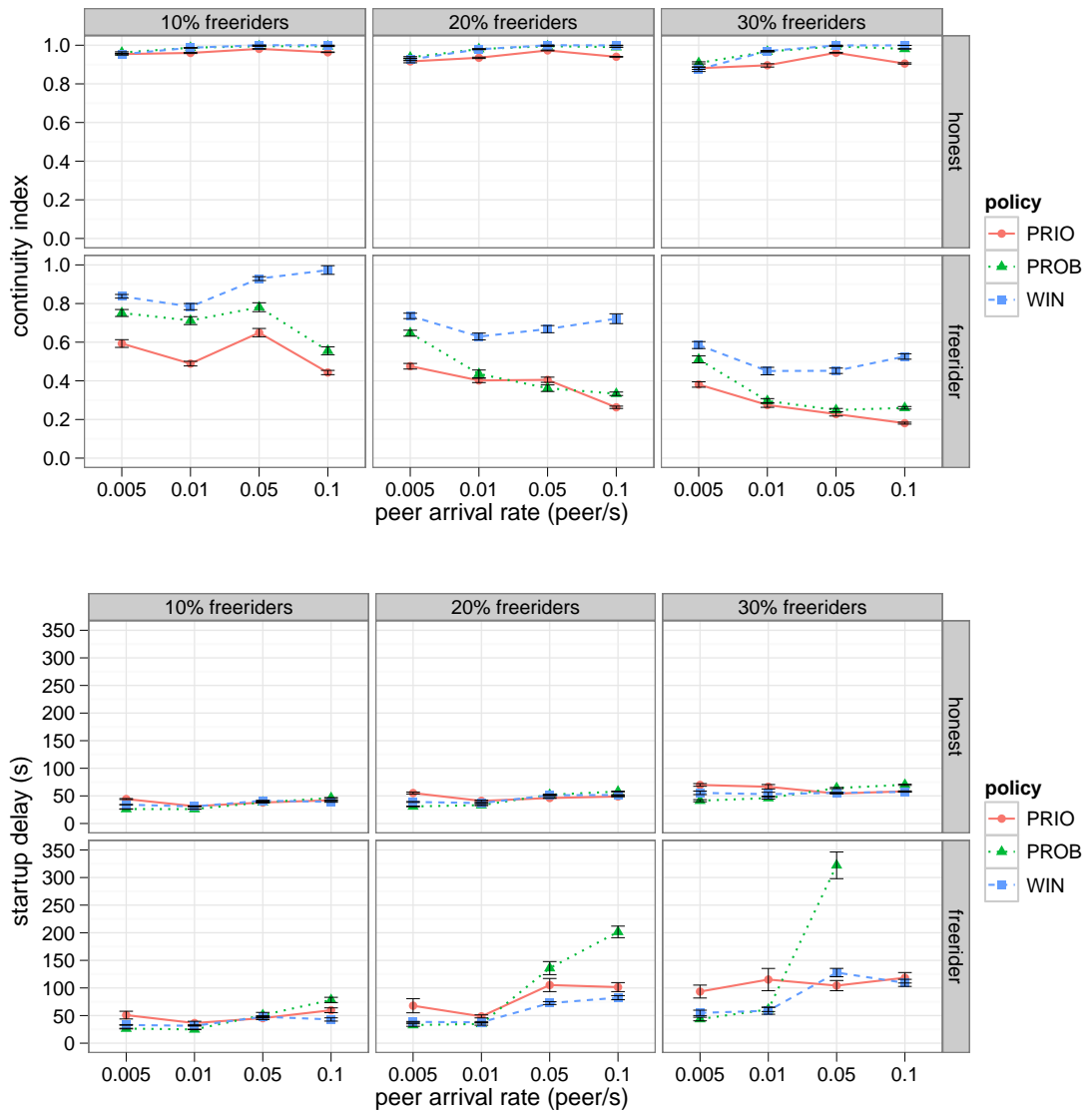


Figure 7: Performance of honest and freeriding peers in a system using indirect reciprocity with different fractions of freeriders.

and the playback progress, which causes peers to miss pieces more frequently. As future work, it might be interesting to see how this policy performs if its download point is redefined to the first piece not yet downloaded, as for the other policies.

Turning our attention to startup delay, we observe that, while for low arrival rates the probabilistic policy provides the shortest startup delays to honest peers, for higher arrival rates the window-based policy performs the best. This is due to the fact that, with the probabilistic approach, peers tend to download also pieces which are farther away. This will increase their bartering ability when peer arrival rate is low but it will slow them down when arrival rate is high. In fact, in the latter case, downloading pieces farther away is unnecessary, since peers have considerably more chance to meet peers with similar level of progress. Finally, we observe that in all cases freeriders experience much longer startup delays when the probabilistic policy is used, thus reducing their incentive to freeride even further.

6.2. Indirect reciprocity

Figure 7 shows the simulation results when using indirect reciprocity. We observe that, in general, this policy provides better QoS to honest peers, both in terms of continuity index and startup delay, than direct reciprocity. At the same time, freeriders experience worse performance. We also observe a lower negative influence of low arrival rates on honest peers. This is due to the fact that peers are rewarded for forwarding, not for bartering. To summarize, we can say that indirect reciprocity works better for VoD than direct reciprocity, for what concerns freeriding resilience.

7. Resilience to Attacks

BitTorrent itself is susceptible to gaming strategies, collusion, and Sybil attacks [23, 29, 21]. Hence, by extension, these attacks can also be performed on BitTorrent-like VoD approaches. In this section, we explore further vulnerabilities of these approaches particularly in the context of VoD. We first develop a better understanding of *who is interested in whom* in these approaches, and then discuss how malicious peers using this insight may formulate strategic attacks that undermine the performance of the system.

7.1. Analyzing Interests

Regardless of the peer selection policy, a VoD-based piece selection policy is based on somewhat in-order downloading of pieces. We hypothesize that peers with lower download progress level typically depend on peers with higher download progress level to upload pieces corresponding to the latter

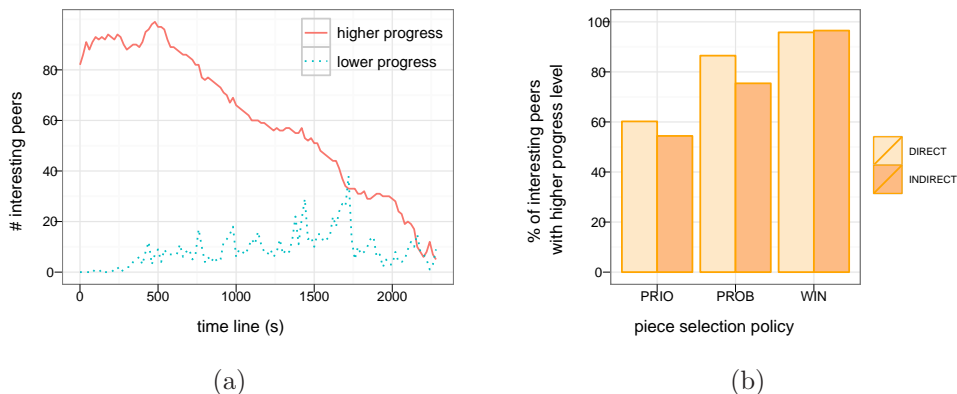


Figure 8: Dependence of peers with higher progress. The peer arrival rate is set to 0.05 peer/s.

part of the file. Put another way, a peer is more often *interested* in peers with higher progress than in those with lower progress.

To verify our hypothesis, we examine which peer is interested in what other peers in the system. Specifically, we measure the number of peers with higher and lower progress that are interesting to a peer at each instance in time. Figure 8(a) illustrates an example of a peer employing probabilistic piece selection policy and direct reciprocity-based peer selection policy. During its lifetime in the system, the number of peers with higher progress are almost always more interesting than those with lower progress. This shows the extent of dependence on peers with higher progress to meet the constraint of playback continuity.

Figure 8(b) plots the percentage of peers with higher progress among all the peers interesting to a peer, across all combinations of piece and peer selection policies. We have used an arrival rate of 0.05peers/s, which is the average value for popular content [31]; the results for other peer arrival rates showed similar results. In each scenario, the majority of the peers that are interesting to a peer have higher progress than itself, confirming our hypothesis. This extent of dependence is more than 95% in the case the window-based piece selection policy, compared to nearly 80% for the probabilistic and less than 60% for the priority-based ones. This is because the latter policies also download pieces that are further ahead in the file.

Another interpretation of this result is that the majority of peers that are interested in a peer p have a lower progress than the peer p itself. This

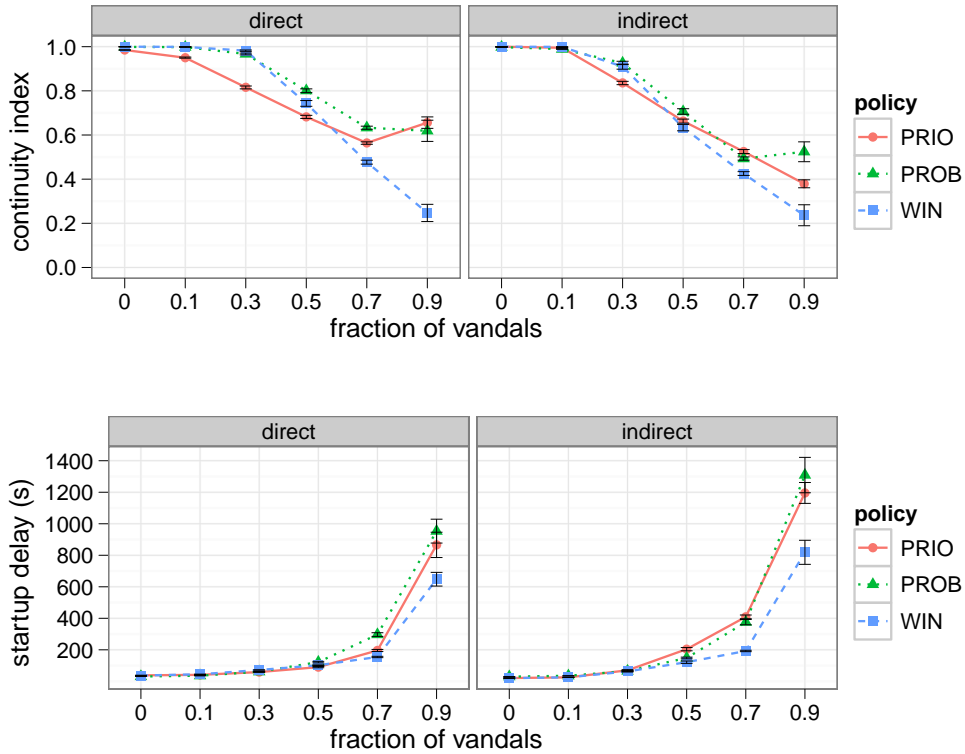


Figure 9: Impact of vandals. The peer arrival rate is set to 0.05 peer/s.

insight motivates malicious peers to throttle honest peers with lower progress by keeping them choked and never uploading. In the remainder of the section, we discuss variants of this strategy and their impact on the performance.

7.2. Vandalism

The main intent of *vandals* is to disrupt the system, even at the cost of not benefiting themselves. Motivated by the above insight, such a vandal acting individually uploads only to peers with higher progress than itself, while choking the rest. Hence, this attack would work even if peers were to use an accounting mechanism to keep track of the contributions of others, while normal freeriders would easily be detected and neutralized.

Figure 9 shows that the average continuity index significantly degrades as the fraction of vandals increases. Even when vandals comprise a mere 30% of all the peers in the system, the performance becomes substandard,

i.e., continuity index is less than 1. In particular, we observe that, among peer selection policies, indirect reciprocity performs generally worse. This is due to the fact that the principle behind this policy is having higher progress peers forward to lower progress ones. Hence, it suffers more from attacks aimed at disrupting this “down-forwarding” chain. For what concerns piece selection policies, the probabilistic one performs better than the other two in terms of continuity index. This is because peers using the former policy also download pieces that are farther from the first missing piece, and hence are lesser dependent on peers with marginally higher progress. With the priority-based piece selection policy, a peer downloads pieces that are farther from the first missing piece only if it is experiencing a good performance. Furthermore, as already observed in Section 6, the fact that the high priority range for the pieces to download is relative to the playback position causes this policy to be more vulnerable to missing pieces. This explains its lower continuity index.

Figure 9 also shows that the average startup delay increases as the fraction of vandals increases. In contrast to continuity index finding, the probabilistic piece selection policy performs worse than window-based and priority-based ones in terms of startup delay. This shows that better playback continuity comes at the expense of longer startup delay. To summarize, the result of this experiment shows that, when more peers adopt the strategy of vandals, it worsens the performance for all peers.

7.3. Collusion

We now describe an attack strategy by a group of malicious colluding peers. Each malicious peer uploads to other malicious peers as well as honest peers with higher progress compared to itself. At the same time, this malicious peer chokes honest peers with a lower progress. This enforces an honest peer to compete with malicious peers for the upload bandwidth of honest peers with a higher progress.

Figure 10 plots the continuity index and the startup delay as we vary the fraction of colluders. Both continuity index and startup delay for honest peers worsen when colluders outnumber honest peers. In contrast, the performance for colluders remains practically unaffected. Among peer selection policies, indirect reciprocity based approaches for honest peers are more resilient to this attack strategy in terms of continuity index. On the down side, it comes at the cost of a longer startup delay. Among piece selection policies, window-based approach performs the best in terms of startup delay while experiencing

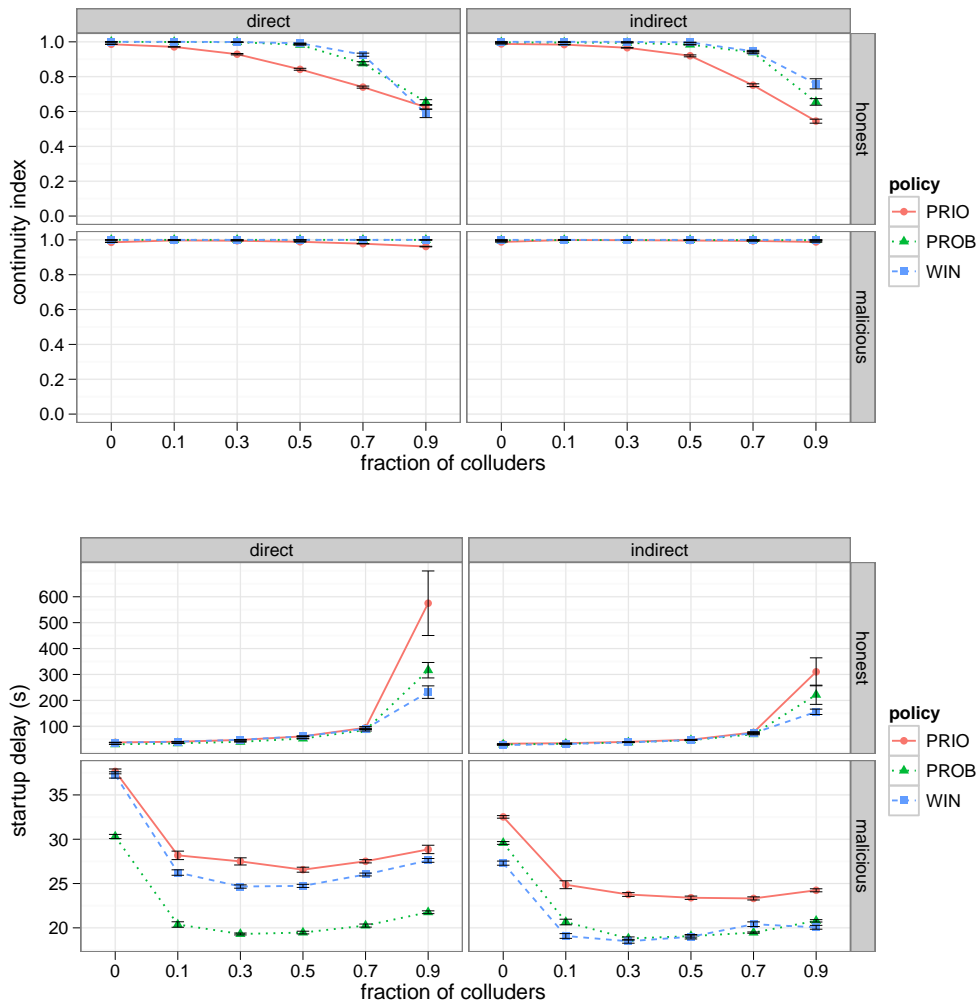


Figure 10: Impact of collusion. The peer arrival rate is set to 0.05 peer/s.

comparable playback continuity.

There are two main implications of this result. First, if a client implementing the strategy of colluders becomes very popular and its peers outnumber those with a default implementation, the latter will experience a substandard performance with continuity index less than 1 and a long startup delay. Second, colluders can start the playback very soon and keep continually playing without any stalling problems during the video, irrespective of the fraction

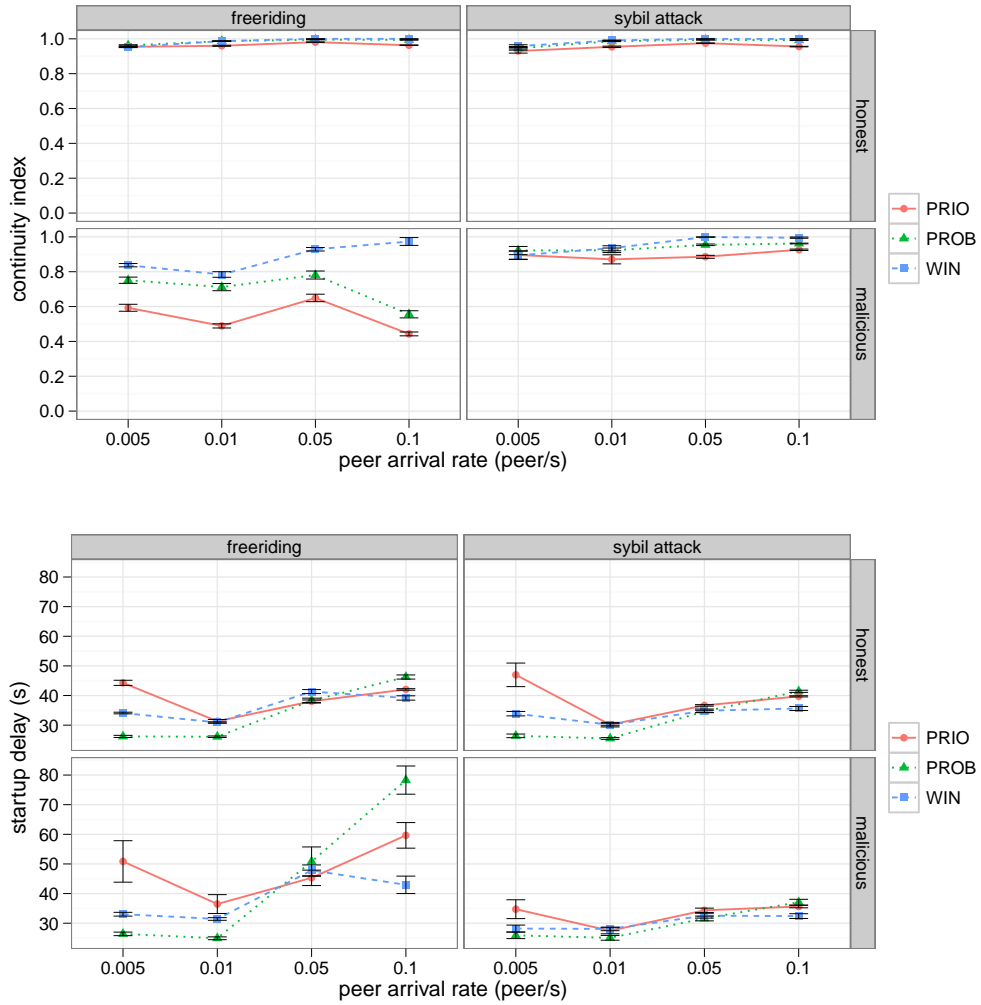


Figure 11: Impact of Sybil attack. The fraction of malicious nodes is set to 0.1 and the peer arrival rate to 0.05 peer/s.

of honest peers.

7.4. Sybil Attack to Indirect Reciprocity

In a system with the peer selection policy based on indirect reciprocity, a peer uploads to those peers that have recently forwarded data to others at the highest rates. To game the system, a malicious peer b first creates Sybils c_1, \dots, c_n which act as b 's children. Whenever b downloads a piece from

another peer a , each of b 's children c_i immediately reports to a that b has forwarded pieces to it at a fast rate. From a 's perspective, b becomes a fast uploader and hence a keeps uploading to b in the future rounds. As a result, the malicious peer b can keep downloading pieces without any incentive to actually forward or upload any data to other peers; in other words, b is a freerider. Since this is a “dominant strategy” for any peer, it results in a “tragedy of commons” if each peer adopts this strategy.

To illustrate the impact of this attack, we compare the performance for honest and malicious peers under the scenarios of freeriding and the proposed sybil attack. Figure 11 plots the continuity index and startup delay as we vary the peer arrival rate. We make the following observations. First, the continuity index for malicious peers employing sybil attack strategy is (i) comparable to that of honest peers and (ii) much better than that for freeriders. Second, the startup delay for malicious peers under sybil attack is (i) lower than that for honest peers and (ii) significantly lower compared to freeriders. This shows that, even without uploading any data to others, malicious peers have a similar performance to that of honest ones. This result highlights the drawbacks of the indirect reciprocity-based approaches which were not considered in [25].

8. Influence of Heterogeneous Peer Bandwidths

The peer selection policies considered in this paper are both based on bandwidth reciprocity: i.e. a peer receives (roughly) as much bandwidth as it provides to the system. In traditional file transfer, this mechanism incentivizes peers to upload as much as they can, in order to get faster downloads. A consequence of this approach is that faster uploaders will download at faster rates and slower uploaders will download at slower rates.

In VoD however, we note that peers do not need to download as fast as possible, but rather maintain a download speed high enough to experience a continuous playback. In other words, a peer's only interest is to maintain a download speed above the video playback rate R . Hence, it is natural to question this approach in the case of peers having *heterogeneous* bandwidths.

In this section, we investigate the performance of reciprocity-based peer selection mechanisms in VoD systems where peers have heterogeneous bandwidths. We consider a scenario where 50% of the peers have a high upload capacity and the other 50% have a slow upload capacity. Although the bandwidths of fast and slow nodes, μ_f and μ_s respectively, are different in each

Table 5: Setups for the heterogeneous scenario.

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

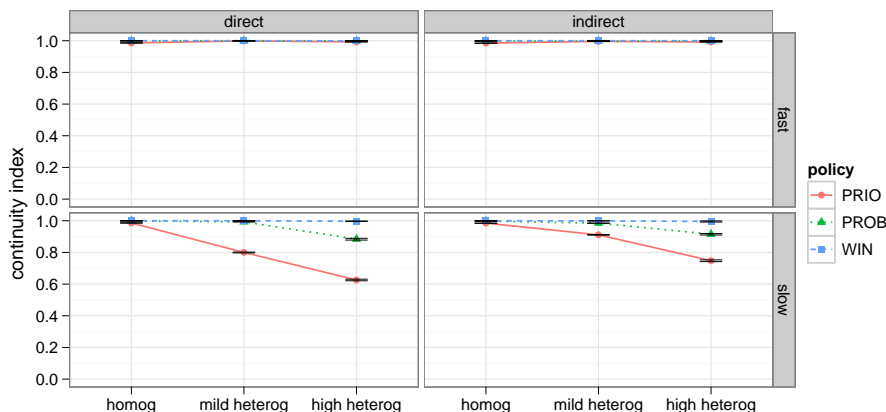


Figure 12: Continuity index in a scenario with heterogeneous peers. The peer arrival rate is set to 0.05 peer/s.

setup, the average peer upload capacity is always equal to $\mu = 1.25R$ (from Table 1). 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 the impact of higher heterogeneity. The homogeneous case, where all peers have an upload capacity $1.25R$, is reported as well for comparison. The details for these setups are shown in Table 5, while all the other simulation settings are as in Table 1 and Table 4.

Figures 12 and 13 plot the continuity index and startup delay of peers, respectively. We note that, as the heterogeneity in the system increases, startup delay for all peers increases while continuity index worsens only for slow peers. This is in line with our previous work [11]. For what concerns peer selection, indirect reciprocity performs better than direct reciprocity, with generally higher continuity index for slow peers and lower startup delay for all peers. Regarding piece selection policies, window-based exhibits the

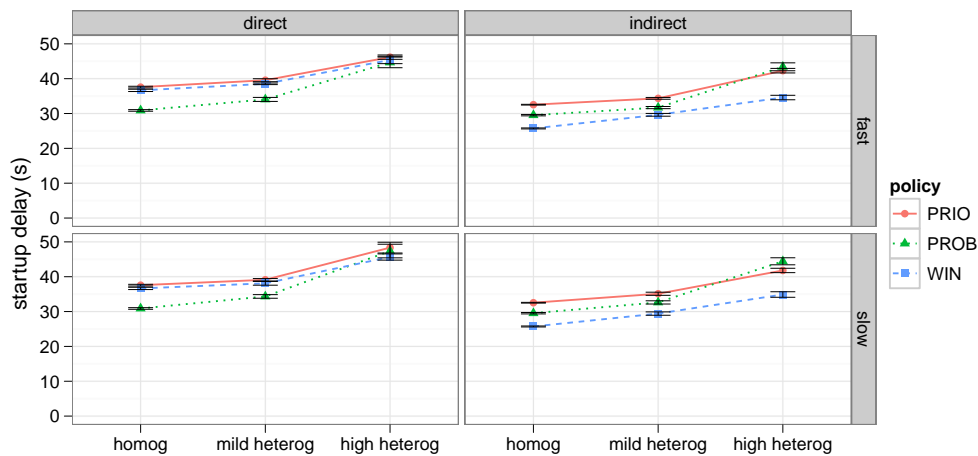


Figure 13: Startup delay in a scenario with heterogeneous peers. The peer arrival rate is set to 0.05 peer/s.

best continuity index and outperforms the other two also for startup delay, at least when used in combination with indirect reciprocity. The more “fair” performance of the window policy is due to the fact that peers only download pieces from a small window ahead the first piece not yet downloaded. Hence, a fast peer cannot barter with other faster peers having a lower or higher progress level, but only with those peers (fast and slow) having similar progress level to itself.

However, the fact that startup delay increases with heterogeneity is an undesired drawback of these VoD approaches. To understand the reasons for this behavior, we have looked at the peers average download rates (Figure 14). The average speed of peers (especially of the fast ones) is generally far above the playback rate R (which is 800 kb/s). In particular, for fast peers, we observe that the download speed is more or less constant (or in some cases growing with the level of heterogeneity) in all setups. Therefore, the increase of startup delay in the heterogeneous scenarios can be explained with peers downloading at slower rates when at the beginning of the file, while downloading much faster when towards the end.

The fact that peers are downloading at rates much higher than needed while others are delayed, means that the bandwidth reciprocity approach is not optimal for the VoD case. Based on this intuition, in a previous work [11]

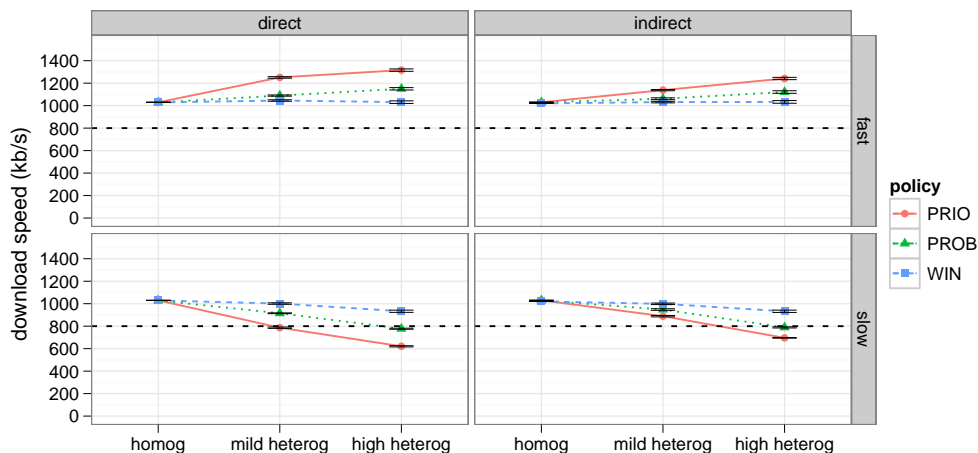


Figure 14: Download speed in a scenario with heterogeneous peers. The peer arrival rate is set to 0.05 peer/s.

we have proposed to equip the reciprocity-based policy with an adaptive “altruism” mechanism. Specifically, our approach is to have a peer utilize the current reciprocity-based approach when its download rate is below or close to the video playback rate R . Conversely, when its download rate is much higher than R , the peer will reduce its reciprocity-based behavior and start allocating bandwidth to peers at random, with a bias towards the new comers. In this way, startup delays are reduced and peer QoS enhanced.

9. Influence of Unconnectable Peers

NATs and Firewalls are well known for causing problems to P2P communication, since peers residing behind those devices are not able to receive inbound connections, unless the NAT/firewall is properly configured.

In this section, we analyze the performance of the current BitTorrent-like VoD approaches in a system where a certain fraction α of nodes are unconnectable. As Skevik *et al.* [34] observed, one of the problems introduced by unconnectable nodes is that they are not reachable by others and thus it is difficult to discover their presence. To improve their discoverability, in our experiments unconnectable peers request new nodes from the tracker more frequently than connectable ones. In this way, more links between the groups of connectable and unconnectable peers are established.

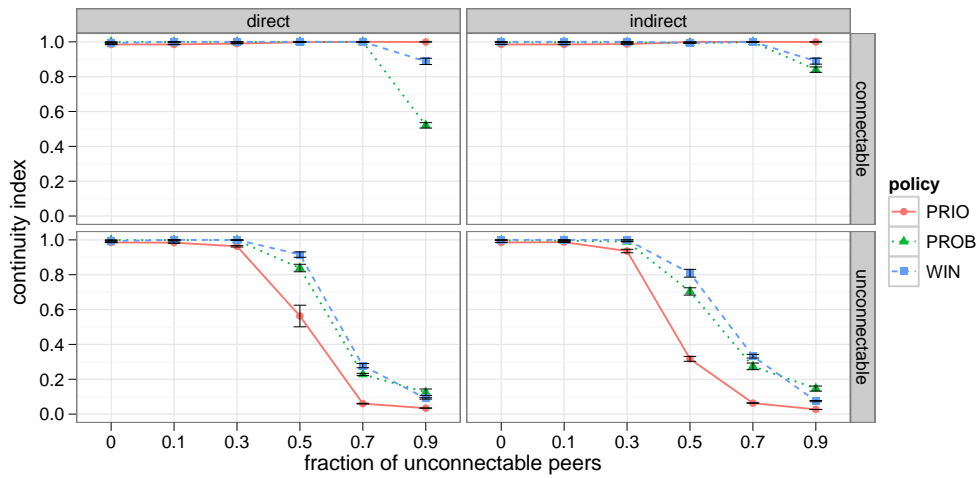


Figure 15: Continuity index experienced by peers when some are unconnectable. The peer arrival rate is set to 0.05 peer/s.

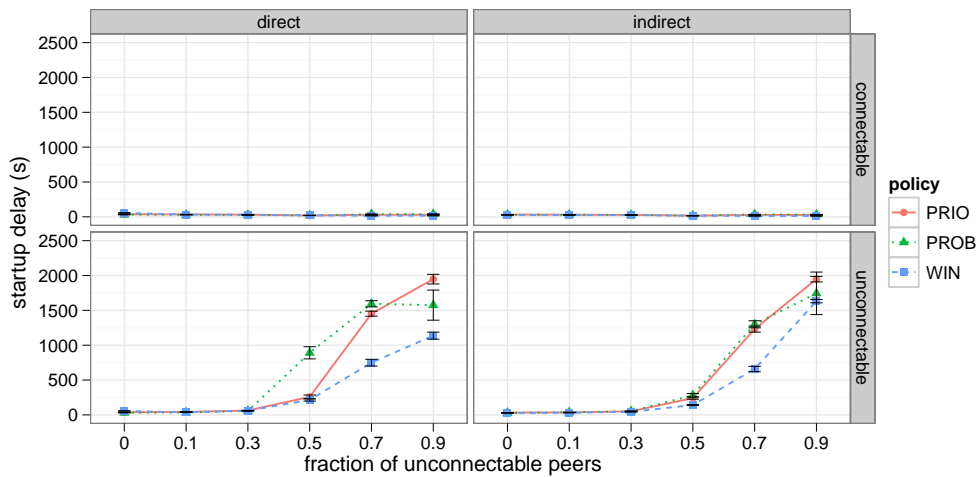


Figure 16: Startup delay experienced by peers when some are unconnectable. The peer arrival rate is set to 0.05 peer/s.

Results for our experiments are depicted in Figures 15 and 16. For what concerns the continuity index, connectable nodes are not affected, until their fraction drops below 10%. The unconnectable nodes, on the other hand, start experiencing a bad continuity index already when their fraction reaches

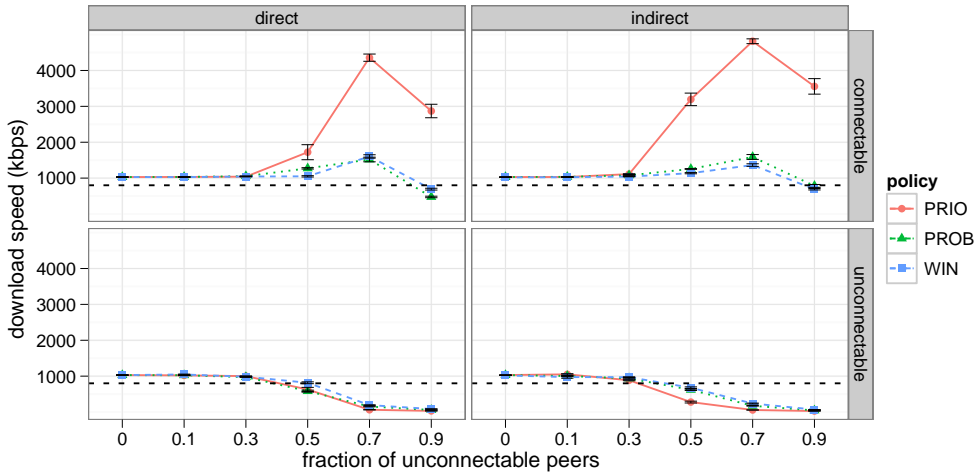


Figure 17: Download speed experienced by peers when some are unconnectable. The peer arrival rate is set to 0.05 peer/s.

50%, which has been measured to be a typical value in BitTorrent as well as BitTorrent-like VoD systems [24, 26]. Similarly, their startup delay suffers a step increase once past the 50% threshold. This phenomenon is due to the fact that unconnectable nodes can only upload to connectable ones, while the latter can potentially upload to any other peer. Hence, when the unconnectable peers are the majority, the connectable ones receive a lot of bandwidth from them while giving only a smaller amount of bandwidth in return. This result was already predicted in our previous work on P2P swarming systems [12]. Consequently, it is clear that the presence of unconnectable nodes severely affects the performance of P2P swarming systems, and VoD systems in particular, and must be kept into account when designing a new P2P system or protocol.

In order to explore avenues for possible improvements, we have looked again at the download speeds of peers (Figure 17). While the download rate of unconnectable nodes drops below the playback rate R (800 kb/s) already when their fraction reaches 50%, that of connectable nodes stays above R , and actually increases, until $\alpha = 90\%$. This implies that connectable nodes often get more bandwidth than necessary to maintain a continuous playback. It is intuitive that this bandwidth surplus could be used to “help” unconnectable peers, in a similar way as for the lower capacity nodes in the

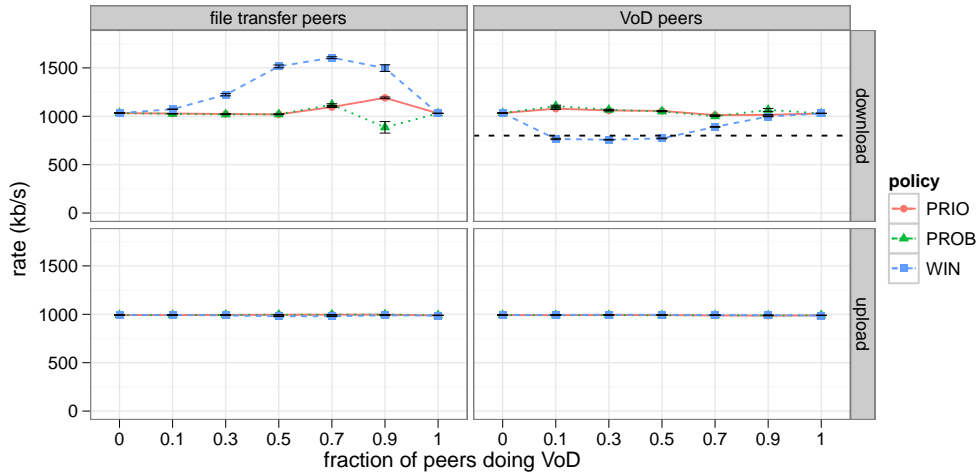


Figure 18: Download and upload rates of nodes doing traditional file transfer against nodes doing streaming, for different VoD piece selection policies (standard BitTorrent is used for file transfer in all cases). The dashed horizontal line in the top-right panel represents the video playback rate R . The peer arrival rate in these experiments is set to 0.05 peer/s.

previous section. The adaptive “altruism” mechanism proposed in our previous work [11] for heterogeneous environments could be used in this case as well, to improve the QoS of unconnectable nodes, without harming that of connectable ones.

10. Coexistence with Traditional File Transfer

In this section, we analyze the implications of having a mixed environment, where peers doing streaming coexist with those doing file transfer using the standard BitTorrent protocol. In fact, nowadays many BitTorrent clients (e.g. BitTorrentDNA [6], μ Torrent [37], and Tribler [36]) already support streaming functionality and more clients are likely to start doing this in the future.

At a first glance, one would expect that introducing VoD nodes in a BitTorrent system where peers do traditional file transfer will negatively affect piece availability and consequently decrease the download speed of the original (non VoD) nodes. In order to verify whether this is really the case, we have performed experiments with increasing fractions of VoD peers. Figure 18 plots the download rates for both file transfer nodes and VoD nodes. As

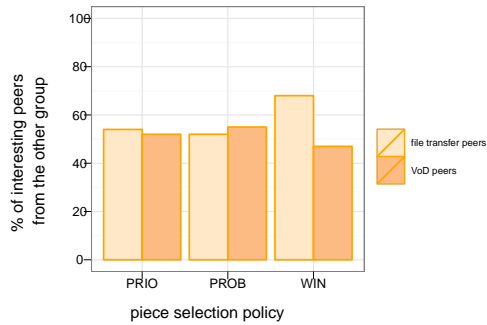


Figure 19: Interest in the other group of peers in a system consisting of both streaming and file transfer nodes.

we can observe, file transfer peers actually *benefit* from the presence of VoD ones: the more the latter, the better for the former. The largest advantage is obtained when the VoD nodes employ the window-based piece selection policy. In order to explain this result, we took a look at the peers interests. Figure 19 shows, for each group of peers, the percentage of interesting peers belonging to the other group, in a scenario where VoD peers are 50% of the total. While for the priority-based and the probabilistic policies the interest of each group in the other one is more or less even, we observe that, for the window-based policy, there are more file transfer nodes interested in VoD nodes (68%) than *vice versa* (47%). We conjecture that the unbalanced interests between the two groups are the main cause of the phenomenon, which can be explained as follows. At first, file transfer nodes download the pieces towards the end of the file (as those pieces are more rare). Later, they need to download the pieces towards the beginning of the file as well, in order to complete their downloads. These pieces are owned mostly by VoD nodes, therefore file transfer peers become very interested in them. On the other hand, VoD nodes using the window-based policy only download pieces within a relatively small window, therefore they will not be very interested in file transfer peers. This unbalance between the two groups' interests causes the VoD nodes to receive many more requests from file transfer nodes than other VoD nodes, which then will be (as a group) optimistically unchoked more often. However, file transfer peers will not have much to upload in return, given the myopic interest of VoD nodes using the window-based policy. As a consequence, file transfer nodes download from VoD nodes much more than they upload to them in return.

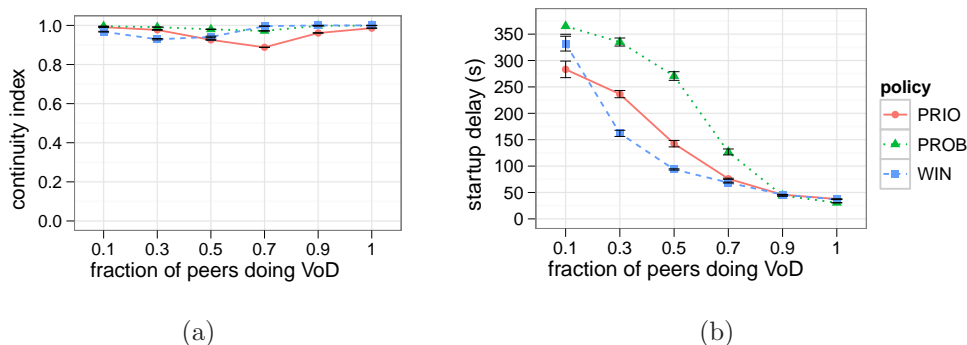


Figure 20: (a) continuity index and (b) startup delay in a system where some peers are doing traditional file transfer while others are doing streaming. The peer arrival rate is set to 0.05 peer/s.

Furthermore, regardless of the piece selection policy employed, VoD nodes suffer from longer startup delays. According to Figure 20, when the fraction of VoD nodes is 0.1, for example, their startup delay is, for all piece selection policies, more than 10 times longer than it would be in a system where all nodes are doing streaming. This is due to the fact that, the more file transfer nodes, the lesser the availability of pieces near the beginning of the file (since file transfer nodes download pieces according to the rarest-first rule). Therefore, it takes longer for VoD nodes to complete the download of the initial pieces necessary for the sequential viewing.

Turning our attention back to the download rates (Figure 18), we see that the priority-based and the probabilistic policies are able to maintain the rates of VoD nodes constantly far above the playback rate (dashed horizontal line in Figure 18). This leaves room for improvement for these two policies. In fact, similarly to the cases of heterogeneous and unconnectable nodes, bandwidth can be allocated in a “smarter” way by having each VoD peer help other VoD ones when its own QoS is good enough.

11. Impact of Watching Behavior

The watching behavior of users in VoD systems can be highly dynamic. For example, it has been shown that users are very impatient and might decide to leave the system after only a few minutes of watching [18, 42]. Similarly, users might want to “jump” to a specific part of the video file. In this section, we analyze how these two types of watching behavior, as

Table 6: Distribution of peer departures according to [42].

departed within (min)	5	10	25	50
% of peers	37.44	52.55	75.25	94.23

observed in real systems, affect the performance of different piece and peer selection policies.

Recall from Section 5.3.3 that the continuity index is defined as the number of pieces received on time divided by the total number of pieces of the file. However, this definition does not make sense for scenarios where peers can leave before file completion or even jump (both forwards and backwards) to another position. Therefore, for the following experiments, we have utilized another definition for the continuity index as

$$CI = \frac{p_w}{p_w + p_m},$$

where p_w and p_m stand for the number of watched and missing pieces, respectively.

11.1. Early departures

From measurement studies over different VoD systems [18, 22, 42], it emerges that the majority of users tends to watch a video for only a few minutes and then leave the system. For example, the distribution presented in [42] shows that at least half of the peers leave within 10 minutes (Table 6). This behavior might have a serious impact on P2PVoD systems, where peers depend on each other’s contribution of bandwidth. To evaluate how well different piece and peer selection policies cope with this problem, we compare a scenario where peers leave only once they have completed the download of the whole video file (standard behavior) against scenarios where they leave earlier. Based on the distribution showed in Table 6, we have created synthetic distributions with different average peer residence times. We denote with T a distribution where 50% of the peers leave within T seconds. Recall from Section 5.2 that the system’s load (i.e. the total number of nodes N) depends on the peer average residence time, m as $N = \lambda m$, where λ is the peer arrival rate. Consequently, to make the comparison among different scenarios fair, for these experiments we have dimensioned the server bandwidth to the different workloads according to Eq. (3):

$$U_s = (\gamma R - \mu) \lambda m,$$

where $\gamma = 1.3$, $\mu = 1000kb/s$, and $R = 800kb/s$ in our setup (see Section 5.3).

Figure 21 shows the results of experiments with peer arrival rate $\lambda = 0.05peers/s$. As we can see, the peers leaving within the average peer residence time experience perfect continuity index in all situations. While the peers remaining longer only experience a minor decrease in their continuity index. The first result can be explained with the fact that, in a P2PVoD system, the piece availability would anyway be skewed towards the pieces at the beginning of the file. Therefore, normally peers proceed fast in downloading the beginning of the file and slower the end. This determines the good performance of peers leaving earlier. On the other hand, since less peers download the ending part of the file, there is less competition for the seeder's bandwidth. As a consequence, the performance of peers staying longer is not significantly affected.

Furthermore, we note that peers leaving earlier has a general beneficial effect over startup delays. This is due to the fact that, since many nodes leave the system after only a few minutes, those that remain are left with fewer peers to upload data to. Consequently, newcomers are more likely to be unchoked earlier and by more nodes, which determines higher initial download speeds and, hence, shorter startup delays.

11.2. *Jumping*

According to recent measurement studies [18, 22], jumping to a different position in a file is not a very common behavior of P2PVoD users and that the number of jumps is also small. Furthermore, both measurement studies [18] and [22] show that jumps, when they occur, are in the form of "random seeking", i.e. the landing position is uniformly distributed across the file and uncorrelated to the current playback position of the user who jumps.

To evaluate the impact of jumps on the QoS, we have performed experiments with growing number of average jumps. The traces analyzed in [18] present an average number of jumps per peer between 1 and 4, therefore we have tested a range going from 0 to 5. The results for the case of arrival rate $\lambda = 0.05peers/s$ are shown in Figure 22. Note that in this type of experiments, besides the initial startup delays, peers are subject to a further delay, which we term *jump delay*, due to the fact that a new set of contiguous pieces have to be downloaded ahead the new playback position, before playback can start again.

From the results of our experiments, we note that the priority-based policy yields to a considerably better performance than the other two piece

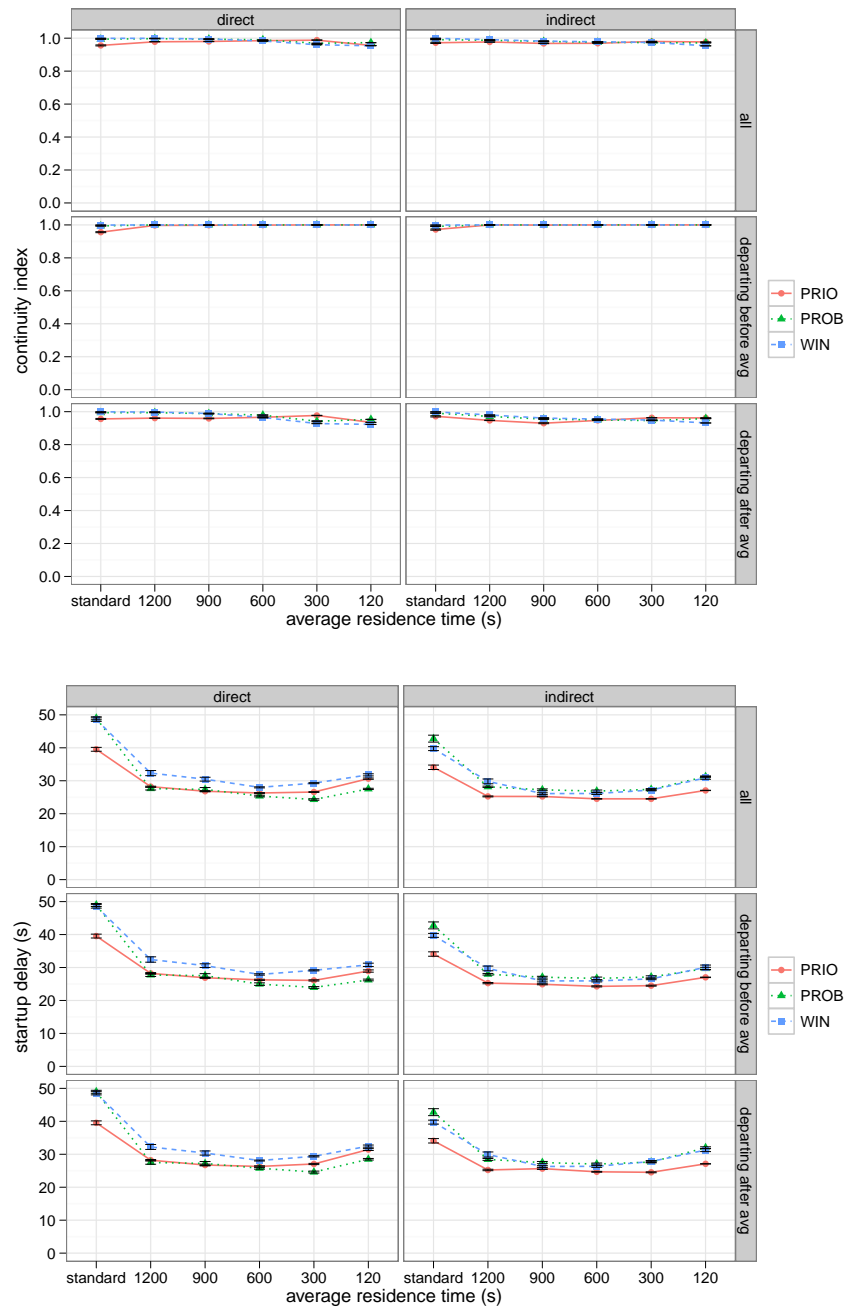


Figure 21: Impact of early departures.

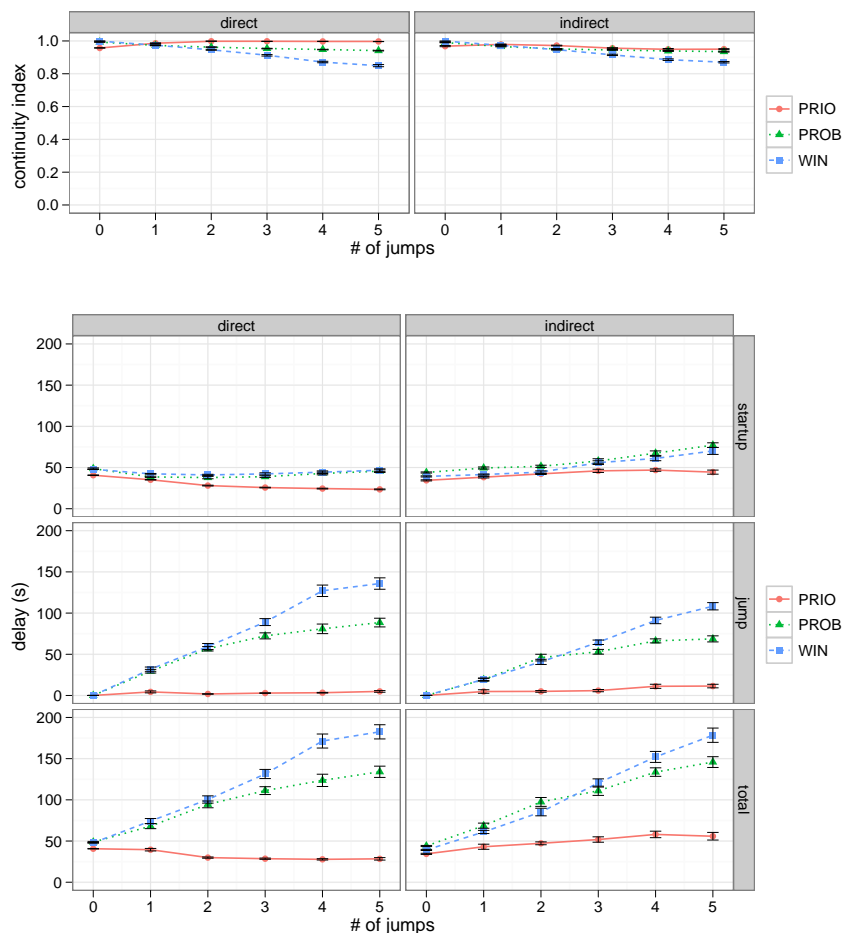


Figure 22: Impact of jumps to a new, uniformly distributed, position in the file.

selection policies, both for what concerns the jump delay and the continuity index. This is due to the fact that, when using this policy, peers also occasionally download pieces according to local rarest-first, which become a precious asset when they jump ahead in the file. In fact, the jumping nodes can immediately barter these rare pieces to get the needed pieces ahead the new playback position. On the contrary, peers using the other two policies do not have many interesting pieces to barter with older peers in the system when jumping ahead, and can only be optimistically unchoked by them. Hence, they experience much longer jump delays.

We also note that, for all piece selection policies, the startup delay experienced with direct reciprocity decreases as we increase the average number of jumps, while with indirect reciprocity the opposite is true. We believe that this phenomenon can be explained as follows. When direct reciprocity is used, a peer b that jumps will try to barter pieces with peers that own the part of the file where b has jumped to. Hence, it will stop uploading to its previous bartering partners and look for new ones. As a consequence of this, b 's previous bartering partners will be left with fewer peers to upload data to, and hence newcomers are more likely to be unchoked earlier and by more nodes, as in the case of early departures. On the other hand, in indirect reciprocity, a peer b receives data from another peer a when b forwards the pieces received from a to others (Figure 1). When b jumps and until it finds better uploaders, it will still continue to forward the pieces received by a to others. Hence, peer a will continue to be busy uploading pieces to b for a while and will not unchoke more newcomers. To verify our hypothesis, we have measured the average time needed by a peer to change 90% of its uploaders after a jump for the case of one jump and piece selection policy WIN. The result of this experiments is that this time is approximately 20s for WIN+DIRECT and 86s for WIN+INDIRECT, hence confirming our hypothesis. Furthermore, since with indirect reciprocity the jumping peers continue to receive some pieces from the initial uploaders, another consequence of this phenomenon is that, at least for WIN and PROB, the jump delay is lower than for the case of direct reciprocity.

12. Discussion

In this paper, we have taken the first steps towards developing a deeper understanding of how the numerous proposed BitTorrent-like P2P approaches to VoD work. On one hand, this allows us to formulate the challenges that future system designers will have to deal with. On the other hand, our study provides useful information that can aid VoD service providers in the selection of the most appropriate protocol for their environment. We shall discuss these two aspects in detail below.

12.1. Implications for future system designers

Rethinking incentives for VoD. We have shown that the current incentive schemes adopted in BitTorrent-like VoD systems are not suitable to VoD. This is due to the fact that current incentives are based on a general file

transfer goal rather than a VoD goal. In file transfer, the goal is to maximize the total download speed, therefore an incentive based on bandwidth reciprocity induces peers to contribute as much as they can. On the other hand, in VoD the goal is to have as many peers as possible experience a smooth playback. In other words, a peer does not earn any benefit in downloading at much higher rates than the playback rate. Based on this insight, future system designers should focus on creating incentives that better fit the VoD case. Finally, it is also useful to observe how the incentives for VoD differ from those for live streaming. In live streaming, peers have a shared temporal focus and therefore a similar playback position. Normally, peers closer to the seeding source will be able to watch the video with a smaller delay than peers further away. Hence, in this case, an effective method to incentivize peers to contribute is to reward high contributors with closer distance from the source, which increases the feeling of “live-ness” of the video. However, this method is not applicable to VoD because of the skewed temporal focus of peers.

More security. We have shown that current approaches are weak against a number of attacks. Future system designers should think of ways to secure them, for instance, by means of some reputation mechanism.

Firewalls and NATs support. We have shown how bad the performance of unconnectable peers can get, even when they are only 30-50% (which is a realistic range in P2P swarming systems, see [24] and references therein) of the total and even when some methods to increase the reachability of these nodes is in place. Therefore, in order to improve their QoS, it is important to make these node completely open, which calls for the need of NAT traversal. This is probably the reason behind the recent development of UDP-based P2P swarming protocols⁶, such as the Peer-to-Peer Streaming Peer Protocol [35] and the uTP protocol in the μ Torrent client [38].

Analyzing scenarios with mixed policies. A system where mixed policies are used can drastically change the performance of these policies. We have evaluated the case where one VoD protocol competes with the original BitTorrent protocol. However, an interesting direction for future work is to test other scenarios too, where, for example, different VoD protocols using different policies compete against each other. The recent paper of Rahman *et al.* [30] is an attempt to give a framework, called Design Space Analysis, for

⁶NAT traversal is much easier to implement in UDP than TCP [5].

testing performance and robustness of P2P swarming systems, which may be applied here.

12.2. Implications for VoD service providers

Our experiments were performed with a constant server bandwidth and then the QoS degradation in the various scenarios was measured. In commercial peer-assisted systems, however, server bandwidth is likely to be dimensioned to the demand (i.e. QoS degradation is compensated by more server bandwidth). Therefore, VoD service providers can interpret our findings as follows: lower QoS means more server bandwidth needed to compensate, which equals to higher costs. An estimation of the needed bandwidth (and therefore the cost incurred) can be directly calculated from the average download rate of peers (for each peer: bandwidth needed equals to playback rate minus average download rate).

Table 7: Summary of findings.

Test case	Direct	Indirect	Overall	Absolute
Freeriding resilience	PROB	PROB	PROB, indirect	●
Resilience to vandalism	PROB	PROB	PROB, direct	—
Resilience to collusion	PRIO	PRIO	PRIO, indirect	◐
Resilience to Sybil attack	all	PRIO	all, direct	●
Heterogeneous bandwidths	WIN	WIN	WIN, indirect	●
Unconnectable peers	WIN	WIN	WIN, direct	◐
Coexistence with trad. file transfer	PRIO	n.a.	PRIO, direct	◐
Early departures	PROB	PRIO	PRIO, indirect	●
Jumps	PRIO	PRIO	PRIO, direct	●

Furthermore, Table 7 summarizes all the experiments done for the tested policies. This table can aid the selection of the best protocol to be used by a VoD service provider, given its particular environment and system characteristics such as open/closed system, with or without unconnectable nodes or nodes with heterogeneous bandwidth, etc. For each test case we selected the best performing piece selection policy for both direct and indirect reciprocity (in the Direct and Indirect columns, respectively). We also show, in the ‘Overall’ column, which policy combination performed the best for each scenario. The last column indicates how well BitTorrent-like VoD performs in the given scenario in absolute terms (full circle means ‘good’, half circle stands for ‘average’, and dash means ‘bad’). When using this table, the

reader should be aware of two things. Firstly, the Sybil attack presented in this paper can only be performed in a protocol using indirect reciprocity, thus all protocols using direct reciprocity are resilient to it. Secondly, traditional BitTorrent works with direct reciprocity only. Therefore, in order to keep the VoD and the file transfer protocols homogeneous, we have focused on the case of direct reciprocity only.

Acknowledgements

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions to improve the quality of the paper. 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. The experiments presented in this paper were performed on the *Distributed ASCI Supercomputer 4* (<http://www.cs.vu.nl/das4>).

References

- [1] E. Adar, B. Huberman, Free riding on gnutella, *First Monday* 5 (10) (2000) pp.2–13.
- [2] S. Annapureddy, S. Guha, C. Gkantsidis, D. Gunawardena, P. R. Rodriguez, Is high-quality vod feasible using p2p swarming?, in: *Proceedings of the 16th International Conference on World Wide Web*, ACM, 2007, pp. 903–912.
- [3] A. Bakker, R. Petrocco, M. Dale, J. Gerber, V. Grishchenko, D. Rabaioli, J. Pouwelse, Online video using BitTorrent and HTML5 applied to Wikipedia, in: *Proceedings of the IEEE P2P*, 2010, pp. 1–2.
- [4] A.R. Bharambe, C. Herley and V.N. Padmanabhan, Analyzing and Improving a BitTorrent Network's Performance Mechanisms, in: *Proceedings of the IEEE INFOCOM*, 2006.
- [5] A. Biggadike, D. Ferullo, G. Wilson and A. Perrig, NATBLASTER: Establishing TCP connections between hosts behind NATs, in: *Proceedings of the ACM SIGCOMM Asia Workshop*, 2005.
- [6] BitTorrentDNA, <http://www2.bittorrent.com/dna>

- [7] Y. Borghol, S. Ardon, N. Carlsson and A. Mahanti, Toward Efficient On-Demand Streaming with BitTorrent, in: Proceedins of the IFIP NETWORKING, 2010.
- [8] N. Carlsson, D. Eager, Peer-assisted on-demand streaming of stored media using bittorrent-like protocols, in: Proceedings of the IFIP NETWORKING, Springer, 2007, pp. 570–581.
- [9] N. Carlsson, D. L. Eager and A. Mahanti, Peer-assisted on-demand Video Streaming with Selfish Peers, in: Proceedins of the IFIP NETWORKING, 2009.
- [10] B. Cohen, Incentives Build Robustness in BitTorrent, in: Proceedings of the Workshop on Economics of Peer-to-Peer Systems, 2003.
- [11] L. D’Acunto, N. Andrade, J.A. Pouwelse and H.J. Sips, Peer Selection Strategies for Improved QoS in Heterogeneous BitTorrent-like VoD Systems, in: Proceedins of the IEEE International Symposium on Multimedia (ISM’2010), 2010.
- [12] L. D’Acunto, M. Meulpolder, R. Rahman, J.A. Pouwelse and H.J. Sips, Modeling and Analyzing the Effects of Firewalls and NATs in P2P Swarming Systems, in: Proceedings of the IEEE IPDPS (HotP2P), 2010.
- [13] P. Garbacki, D. Epema, J. Pouwelse, M. Van Steen, Offloading servers with collaborative video on demand, in: Proceedings of the 7th International Workshop on Peer-to-peer systems, USENIX Association, 2008.
- [14] Y. Guo, K. Suh, J. Kurose, D. Towsley, P2cast: peer-to-peer patching scheme for vod service, in: Proceedings of the 12th International Conference on World Wide Web, ACM, 2003, pp. 301–309.
- [15] A. Habib, J. Chuang, Service Differentiated Peer Selection: An Incentive Mechanism for Peer-to-Peer Media streaming, *IEEE Transactions on Multimedia* 8 (2006) 610–621.
- [16] X. Hei, C. Liang, J. Liang, Y. Liu and K.W. Ross, A Measurement Study of a Large-Scale P2P IPTV System, *IEEE Transactions on Multimedia* 9 (2007) 1672 – 1687

- [17] C. Huang, J. Li, and K. Ross, Can Internet Video-on-Demand Be Profitable, in: Proceedings of the ACM SIGCOMM, 2007.
- [18] Y. Huang, T. Fu, D. Chiu, J. Lui, C. Huang, Challenges, design and analysis of a large-scale p2p-vod system, ACM SIGCOMM Computer Communication Review, 38 (4) 2008 pp. 375–388.
- [19] M. Izal, G. Urvoy-Keller, E. Biersack, P. Felber, A. Al Hamra, L. Garcés-Erice, Dissecting bittorrent: Five months in a torrent’s lifetime, in: Passive and Active Network Measurement, Springer Berlin / Heidelberg, 2004.
- [20] A. Legout, G. Urvoy-Keller and P. Michiardi, Rarest First and Choke Algorithms are Enough, in: Proceedings of the ACM IMC, 2006.
- [21] D. Levin, K. LaCurts, N. Spring, B. Bhattacharjee, Bittorrent is an auction: analyzing and improving bittorrent’s incentives, in: Proceedings of the ACM SIGCOMM, 2008.
- [22] C. Li, C. Chen, Measurement-based study on the relation between users’ watching behavior and network sharing in P2P VoD systems, Computer Networks, 54(1) 2010 pp. 13–27.
- [23] T. Locher, P. Moor, S. Schmid, R. Wattenhofer, Free riding in bittorrent is cheap, in: Proceedings of HotNets-V, 2006.
- [24] M. Meulpolder, L. D’Acunto, M. Capota, M. Wojciechowski, J.A. Pouwelse, D.H.J. Epema and H.J. Sips, Public and private bittorrent communities: A measurement study, in: Proceedings of the IPTPS, 2010.
- [25] 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: Proceedings of the SPIE MMCN, 2008.
- [26] J.J.D. Mol, A. Bakker, J.A. Pouwelse, D.H.J. Epema, and H.J. Sips, The Design and Deployment of a BitTorrent Live Video Streaming Solution, in: Proceedings of IEEE International Symposium on Multimedia (ISM’09), 2009.

- [27] P2P Next Community CDN for Video Distribution, <http://blog.wikimedia.org/2010/09/27/video-labs-p2p-next-community-cdn-for-video-distribution/>
- [28] N. Parvez, and C. Williamson and A. Mahanti and R. Carlsson, Analysis of BitTorrent-like Protocols for On-Demand Stored Media Streaming, in: Proceedings of the ACM SIGMETRICS, 2008.
- [29] M. Piatek, T. Isdal, T. Anderson, A. Krishnamurthy, A. Venkataramani, Do incentives build robustness in bittorrent?, in: Proceedings of the NSDI'07, Cambridge, MA, 2007.
- [30] R. Rahman, T. Vinkó, D. Hales, J. Pouwelse, H. Sips, Design space analysis for modeling incentives in distributed systems, in: Proceedings of the ACM SIGCOMM, 2011, pp. 182–193.
- [31] B. Zhang, A. Iosup J.A. Pouwelse, D.H.J. Epema, Identifying, Analyzing, and Modeling Flashcrowds in BitTorrent, P2P11, 2011.
- [32] P. Savolainen, N. Raatikainen and S. Tarkoma, Windowing BitTorrent for Video-on-Demand: Not All is Lost with Tit-for-Tat, in: Proceedings of the IEEE GLOBECOM, 2007.
- [33] P. Shah and J. F. Paris, Peer-to-Peer Multimedia Streaming using BitTorrent, in: Proceedings of the IEEE IPCCC, 2007.
- [34] K. Skevik, V. Goebel, T. Plagemann, Evaluation of a comprehensive p2p video-on-demand streaming system, Computer Networks 53 (4) (2009) pp. 434–455.
- [35] Peer-to-Peer Streaming Peer Protocol (PPSPP), <http://datatracker.ietf.org/doc/draft-ietf-ppsp-peer-protocol/>
- [36] Tribler, <http://www.tribler.org>
- [37] μ Torrent, <http://www.utorrent.com>
- [38] uTorrent uTP Documentation, <http://www.utorrent.com/help/documentation/utp>
- [39] A. Vlavianos, M. Iliofotou and M. Faloutsos, BiToS: Enhancing BitTorrent for Supporting Streaming Applications, in: Proceedings of the IEEE Global Internet Symposium, 2006.

- [40] C. Wu, B. Li and S. Zhao, Multi-Channel Live P2P Streaming: Refocusing on Servers, in: Proceedings of IEEE INFOCOM, 2008.
- [41] Y. Yang, A. Chow, L. Golubchik, D. Bragg, Improving QoS in BitTorrent-like VoD systems, in: Proceedings of the IEEE INFOCOM, 2010.
- [42] H. Yu, D. Zheng, B.Y. Zhao, W. Zheng, Understanding User Behavior in Large-Scale Video-on-Demand Systems, in: Proceedings of the ACM EuroSys, 2006.
- [43] M. Zghaibeh, K.G. Anagnostakis, F.C. Harmantzis, The behavior of free riders in bittorrent networks, in: Handbook of Peer-to-Peer Networking, Springer, 2010, pp. 1207–1230.
- [44] Dabek, Frank and Li, Jinyang and Sit, Emil and Robertson, James and Kaashoek, M. Frans and Morris, Robert, Designing a DHT for low latency and high throughput, In Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation - Volume 1, 2004
- [45] Stefan Saroiu, P. Krishna Gummadi, and Steven D. Gribble. A measurement study of peer-to-peer file sharing systems, In Proceedings of Multimedia Computing and Networking 2002 (MMCN '02), San Jose, CA, January 2002.
- [46] Gummadi, Krishna P., Saroiu, Stefan and Gribble, Steven D. King: estimating latency between arbitrary Internet end hosts, Proceedings of the 2nd ACM SIGCOMM Workshop on Internet Measurement, 2002

Appendix

Our mathematical investigation aims to characterize and understand the bartering ability of peers and its relation to the fundamental properties of BitTorrent-like VoD systems. Bartering ability of a given peer i is expressed by the expected value of the number of peers in the system with which peer i can exchange pieces of the file.

In the following, we analyze the window-based piece selection policy. First, we derive the average number of *potential* bartering partners N_b for

any peer in the system. In order to do so, we notice that each peer i joins $1/\lambda$ seconds after its predecessor $i - 1$ to the system. Assuming that each peer i downloads pieces at a rate at least equal to the playback rate R , then i is, on average, $b_{\min} = R/(\lambda P_s)$ pieces behind its predecessor $i - 1$, where P_s is the size (in Kb) of a piece. This implies that a peer has an overlapping window with $\lceil \frac{w}{b_{\min}} \rceil - 1$ many of its predecessors. The same holds for the successors of peer i . Therefore, the average number of potential bartering partners for any peer i is

$$N_b = 2 \left(\left\lceil \frac{w}{b_{\min}} \right\rceil - 1 \right) = 2 \left(\left\lceil \frac{w P_s}{R} \lambda \right\rceil - 1 \right).$$

This concept is exemplified in Figure 23.

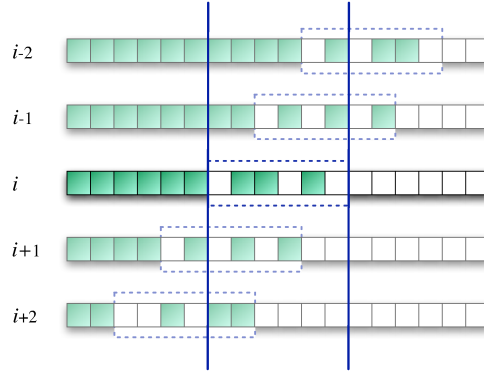


Figure 23: Example of overlapping windows among consecutive peers. Peers are numbered according to the time they joined the system. $w = 6$ and $\lambda = \frac{R}{2P_s}$, which implies that each peer i is 2 pieces behind its predecessor and thus has $N_b = 2(\frac{wP_s}{R}\lambda - 1) = 4$ bartering partners.

This simple analysis shows that a low arrival rate can easily lead to $N_b = 0$, thus giving no chance for bartering, if the system parameters R , P_s and w are not set up carefully. Furthermore, it is desired in VoD systems that $N_b \geq \nu$ holds, i.e., the number of potential bartering partners should be more than the number of upload slots of the peers, otherwise peers would start allocating upload slots to randomly chosen neighbors making the system less resilient to freeriding. Note that it is not given that a peer can actually barter with N_b many peers, that needs to be analyzed below. However, we can already see that N_b depends on the arrival rate λ and the sequentiality parameter

w , thus it can be adjusted to be high enough (meaning high probability of possible piece exchange between peers in the system), but that can lead to longer startup delays. Hence, we argue that *adopting a reciprocity-based approach, in a VoD system characterized by low peer arrival rates, implies that a trade-off exists between freeriding resilience and peer QoS.*

Knowing the number of potential peers for bartering, we are interested now in the probability that two peers can exchange pieces between each other, assuming they have overlapping windows. For the window-based piece selection policy we can assume that both peers have downloaded half of the pieces of their current window, and that peer i is behind peer j in $d \geq b_{\min}$ pieces. Using the fact that the probability that two peers, i and j , cannot exchange any piece is the same as the probability that peer i cannot give pieces to peer j , we obtain

$$\begin{aligned} P[\text{two peers can exchange}] &= \\ &= 1 - P[\text{peer } i \text{ cannot give piece to peer } j] = \\ &= 1 - \sum_{k=\max\{0, d+1-w/2\}}^{\min\{d-1, w/2\}} \binom{d-1}{k} \binom{w-d-1}{w/2-k} \binom{w/2+k-1}{k} B^*, \end{aligned}$$

where $B^* = \binom{w-1}{w/2}^{-2}$ and we assume that $w > d$ holds. In this sum, the first term counts the number of cases that pieces can be located within the first d slots at peer i , the second term is the number of cases that pieces can be located within the overlapping area of peer i and peer j , the third term is the number of cases that the still missing pieces of peer j can be located, and finally $1/B^*$ counts the total number of possible cases. As this probability depends on w and on d , in the following we are using the notation $\mathcal{P}(w, d)$ for it.

Note that $\mathcal{P}(w, d)$ is monotonically decreasing with the increase of d ; it stays very close to 1 up to $d = w/2$ and starts decreasing quickly from there on.

Using the number of potential bartering peers N_b and the probability $\mathcal{P}(w, d)$ that two overlapping peers can barter, we are ready now to calculate the expected value of the number of peers with which peer i can barter, that is

$$E[\text{number of bartering partners of } i] = 2 \sum_{d \in D} \mathcal{P}(w, d),$$

where $D = \{b_{\min}, \dots, w-1\}$. Here, for the sake of simplicity, we assume

that the probabilities that peer i can barter with its potential partners are independent from each other.