European Journal of Scientific Research
ISSN 1450-216X Vol.58 No.4 (2011), pp.550-569
© EuroJournals Publishing, Inc. 2011
http://www.eurojournals.com/ejsr.htm

BitTorrent-Like Protocols for Interactive Access to VoD Systems

Luiz J. H. Filho
Faculdades Integradas Camões
Rua Jaime Reis, 531A - Alto São Francisco - Curitiba – Paraná – Brazil
E-mail: jhfilho@land.ufrj.br

Carlo K. S. Rodrigues
Faculdade de Tecnologia e Ciências Sociais Aplicadas – FATECS
Centro Universitário de Brasília – UniCEUB
SEPN 707/907 – CEP 70790-075 – Brasília – DF – Brazil
E-mail: carlokleber@gmail.com

Rosa M. M. Leão
Federal University of Rio de Janeiro – UFRJ
System Engineering and Computer Science Program – COPPE/PESC
Mail Box 68511 – Zip Code 21941-972 – Rio de Janeiro – RJ – Brazil
E-mail: rosam@land.ufrj.br

Abstract

This article presents two novel protocols for interactive access to video-on-demand systems. They are both based on the well-known BitTorrent paradigm. As main innovative aspects, the video chunks are categorized in different priority sets and there is the deployment of a user-behavior predicting model. The analysis and the validation are carried out through simulations using workloads from a real server. Furthermore, the novel protocols are compared with others of the literature. The final results outline optimizations of up to one order of magnitude over the various competitive metrics considered herein.

Keywords: BitTorrent protocol, Hidden Markov Model, Multimedia, Streaming.

1. Introduction

Numerous solutions for improving video-on-demand (VoD) system scalability have been proposed in the Literature (e.g., [19, 20]). Within this context, peer-to-peer (P2P) architectures have been investigated lately (e.g., [13, 14]). In this type of architecture, clients work collaboratively with the server by helping it at the delivery of the multimedia object over the entire network. There is no need for multicast service, content distribution networks or proxies. This far optimizes the overall system complexity. Besides, server bandwidth requirements are notably reduced [10, 13, 14].

Among the P2P solutions (e.g., [22, 23, 25]), those based on the adaptation of the BitTorrent protocol [4] deserve special attention. They consider the problem of adapting BitTorrent to support on-demand streaming and, more specifically, a view-as-you-download service. From a user experience point-of-view, the system provides low latency to begin playback as well as jitter-free playback [1].

The efficiency of the BitTorrent protocol has already been confirmed in previous work [17]. It is very adequate especially for object replication in P2P networks based on mesh architectures. This protocol divides the object into chunks which are delivered in an out-of-order manner. A client, also denoted as a node, may retrieve distinct chunks of the object from different nodes simultaneously. The total delivery time of the whole object may be then significantly shorter than that when the object is retrieved from a single server.

The main challenge for adapting the BitTorrent protocol basically relies on two points. Firstly, the chunks must be received by the clients within a given time. Otherwise, it will be useless since client will be expecting to receive chunks ahead of those then received. However, as the chunks are not requested in-order, we may not guarantee this time constraint. Secondly, the protocol makes use of particular politics (tit-for-tat) to provide incentives for clients to collaborate at the object delivery. The outcome is that the first clients to be served by the system are those who have already collaborated at the object delivery; in other words, they will have a certain priority to download chunks from other clients. On one hand, this eliminates the so-called selfish clients but, on the other hand, this procedure may negatively impact on the time interval the client will have to wait until he may be then served, the so-called latency.

Within this context, the main contribution of this article is the proposal of two novel protocols for VoD systems with interactive clients: the BitTorrent Interactive Protocol - The first (BIP-F) and the BitTorrent Interactive Protocol - The second (BIP-S). Both proposals are designed for interactive multimedia streaming in mesh architectures. Their conceptions are mainly based on the definition of chunk sets and the deployment of a user-behavior model to forecast the chunks that will be requested in the future. The sets are dynamically updated in function of the client's interactive actions during the play. The same politics of the BitTorrent protocol is used to select the nodes from whom the chunks are retrieved.

The validation of the novel proposals is carried out through simulations using workloads generated from real user-logs of a multimedia server (RIO - Random I/O System) [5]. Distinct performance metrics are considered and competitive analyses are carried out considering other proposals of the literature. The final results indicate optimizations of up to one order of magnitude.

The remainder of this text is organized as follows. In Section 2, we briefly comment on P2P-network structures for video delivery as well as explain how the BitTorrent protocol operates. Besides, we also present the user-behavior model [7] deployed in our proposals and talk about the most popular design strategies for adapting BitTorrent to VoD service. The goal of this section is to provide the basis for a thorough understanding of this text. Section 3 presents the most recent protocols devoted to the adaption of the BitTorrent protocol to the video-delivery service [22, 23, 25, 8]. These proposals are in fact the state-of-art of this research field and are all considered in the competitive analysis included in this work. Our proposals are thoroughly explained in Section 4. In Section 5, we have the performance evaluation experiments which are carried out through simulation. At last, conclusions and future work are included in Section 6.

2. Fundamentals

2.1. Peer-to-Peer Structures

There are basically three structures for P2P video delivery: the tree structure; the mesh structure; and the hybrid structure. Under the tree structure (Figure 1 (a)), the nodes (peers) are organized into a single tree or into multiple trees. The video source is the tree root. Hierarchical relationships among the nodes are formed: the parent node is the only one to send video streams to his children nodes. The works of [3, 16] are examples of video-delivery proposals that make use of tree structures.

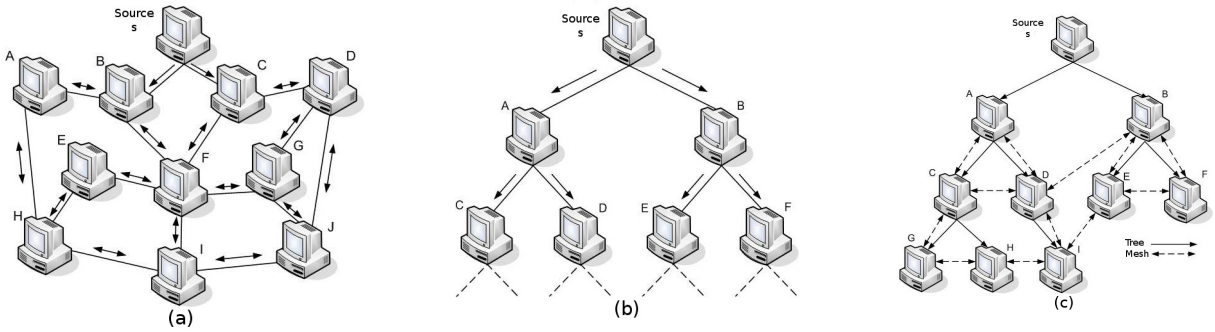
In the mesh structure (Figure 1 (b)), the nodes constitute an overlay network and deliver the video in chunks. There is no hierarchy among the nodes, i.e., any node may receive or transmit the

chunks to any other node in the network. The works of [8, 22, 23, 25], make use of the mesh structure for video delivery.

Finally, the combination of the tree and mesh structures form up the hybrid proposal (Figure 1(c)). It tries to capture the benefits of both proposals previously mentioned. However, the side effect is that its implementation is rather complex. The works of [18, 24] propose hybrid structures for video delivery.

Both the BIP-F and the BIP-S protocols make use of the mesh structure. This is due to the fact that this structure has proved to be the most simple to be implemented as well as the most stable one [22, 23, 25, 8].

Figure 1: Video-delivery structures in P2P networks: (a) Tree; (b) Mesh; (c) Hybrid.



2.2. BitTorrent Protocol

In the BitTorrent protocol there are two types of nodes (peers): leechers and seeds. Leechers are nodes that do not have all the object chunks, while seeds are nodes that have all chunks. Seeds only transmit chunks, while leechers transmit and receive chunks.

A swarm is a set of nodes that take part in the transmission and reception of a same object. Each swarm is controlled by a central process denoted as tracker. To join a swarm, it is necessary that the leecher contacts the tracker. The tracker then passes the leecher a list of nodes that have the wanted object. The leecher then selects a subset of the list and tries to establish TCP bidirectional connections. The nodes of the subset are denoted as *neighbors*.

The BitTorrent protocol also has an important incentive strategy (tit-for-tat) to avoid the free-riding process, that is, the condition that selfish nodes just use the swarm to receive object chunks and never transmit chunks to any other node. Under this strategy, each node typically transmits to the k nodes that recently have provided him with the best download rates, even though there may be more than k nodes interested in receiving chunks of the object.

The refusal to transmit to certain nodes, intrinsic to the *neighbor (or peer) selection policy*, is denoted as a *choking process*. The *optimistic unchoking* is the process that allows clients to reserve part of their bandwidth to transmit to nodes randomly chosen. The choking and unchoking processes are done periodically.

Still to improve the swarm efficiency, the BitTorrent protocol makes use of a rarest-first policy to select the object chunks that will be retrieved. The first chunks are those that are less replicated. Another important characteristic, which increases the download rate, is that the chunks are divided into subchunks which may be requested from different nodes simultaneously. This is all called the *chunk (or piece) selection policy*.

In a few words, the BitTorrent protocol has a low control overhead, is scalable, efficient and easy to implement. Content replication is its main goal. However, it may not be directly deployed for on-demand streaming without taking account the particularities just outlined in Section 1.

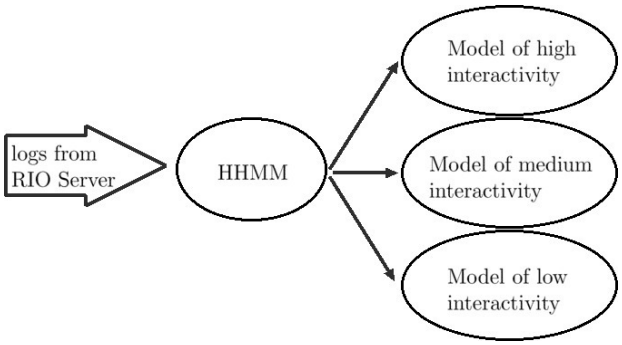
2.3. An User-Behavior Model

This section briefly discusses the user-behavior model presented in the work of [7]. It constitutes a Hierarchical Hidden Markov Model (HHMM) to emulate the behavior of clients accessing a multimedia server. This model was based on a real application of distance learning, where students from the CEDERJ [5] course attend classes, that were previously recorded and synchronized with slides [6], using the RIO Server [5].

The model's hierarchical structure has several interesting properties: the training phase complexity is smaller than that of the conventional HMM, the short-term dependencies are captured by the lower-hierarchy chain, and the long-term dynamics is governed by the hidden markov chain (higher-hierarchy chain). An important aspect of the operation of our proposals, namely BIP-F and the BIP-S protocols, is their integration to this user-behavior model, i.e., how this model is used by our proposals, as briefly explained next.

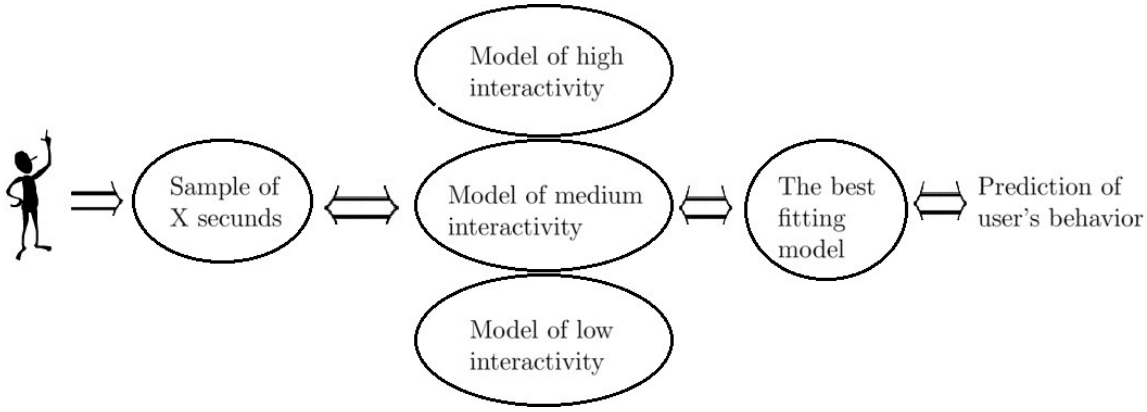
The logs captured from the RIO server [5] are categorized in function of the user-interactivity level, which is a function of the number of interactivity actions the user has done. This has resulted in three distinct categories: 16 – 40 (high); 6 – 15 (medium); 0 – 5 (low). A user model is then generated and trained for each interactivity level. This idea is depicted in Figure 2.

Figure 2: Categorization of the user-interactivity level.



Each of these models emulates a sequence of interactive actions, with parameters set in function of the real user logs. The sequence is the input to the experiments with BIP-F and BIP-S protocols and works as the prediction of the client's future interactive actions. This whole procedure is done off-line.

Figure 3: Determination of interactivity level.



When a new user arrives in the system, it is necessary to determine to which interactivity level he belongs to. The only available information is: the video he wants to watch and his interactivity actions as soon as the play starts, which are stored in a log file. To determine the interactivity level a

sample of length *X* seconds is examined. This sample is used as an input to each of the three user-behavior models (high, medium and low) and the maximum likelihood function is computed in order to determine the best fitting model. This idea is depicted in Figure 3.

To end this section we point that, for the sake of generality and simplicity, in this text we refer to this model as *user-behavior model* or *user-behavior predicting model* interchangeably.

2.4. Design Strategies

The adaptation of the BitTorrent protocol to be used in P2P VoD systems is mainly based on modifications of the chunk and/or neighbor selection policies [1, 15].

Related studies on the chunk selection algorithms have concentrated on finding a policy that can achieve a good trade-off between meeting the sequential requirement of playback and maintaining a high level of chunk diversity in the system. The most important prior research can be broadly categorized into three design strategies: (i) probabilistic [2, 9]; (ii) windows-based [21, 22, 23, 25]; (iii) adaptative-window [1].

Considering the first strategy, chunks are selected in function of some probability distribution function. Under the second strategy, a sliding window of fixed size is typically used and chunks within it are given preference. Lastly, considering the third strategy, each peer dynamically computes its window size, depending on how well the peer's download is progressing.

Now judging from the studies of neighbor selection policies so far presented in the literature, we may notice that the original choke algorithms of the BitTorrent protocol are still the most competitive alternatives [2, 22, 17]. The proposals of modifications still seem to be at an initial stage of development. The proposal of [22], for instance, uses a randomized policy in which, at the beginning of every playback, each peer selects neighbors at random for a *randomized choking interval*. These neighbors are selected from the list of peers received from the tracker. This gives more free tries to a larger number of peers in the swarm to download chunks. The peers will possibly share these chunks later (see Section 3.2 for more details).

To the best of our knowledge, so far there have not been yet any interactivity-service proposals purely based on the Bittorrent-like paradigm (e.g., [11, 12]). For example, the work of [11] makes use of a P2P mesh-architecture together with a centralized server. And the work of [12] deploys a tree-architecture to deliver the media object with interactivity.

We finally outline that both the BIP-F and the BIP-S protocols are mainly based on modifications of the chunk-selection policy. This is due to the fact that this way has so far revealed to be more productive and simpler (e.g., [22, 23, 25, 8]).

3. Recent Proposals

Herein we present the most recent protocols devoted to the adaption of the BitTorrent protocol to the video-delivery service (e.g., [8, 22, 23, 25]). In spite of not being designed for an interactivity service, these proposals are some of the state-of-art algorithms belonging to this research field are all considered in the competitive analysis carried out in Section 5.

To have a fair comparison between these proposals and the novel protocols to be presented in this work, in the performance evaluation experiments of Section 5, we made several adaptations so that these past proposals could properly meet the interactivity requirements. These adaptations are implicit and mainly consist of making the chunk sets dynamic. This means that the sets may be then updated every time the play point of the object is changed due to a user's interactivity action.

3.1. BiToS and Zhou-Chiu-Lui

The BitTorrent Streaming Protocol (BiToS) [23] is devoted to a sequential visualization (in-order) of the object chunks, i.e., there is no interactivity. This proposal differs from the original BitTorrent protocol exclusively with respect to the chunk-selection policy, as we explain next.

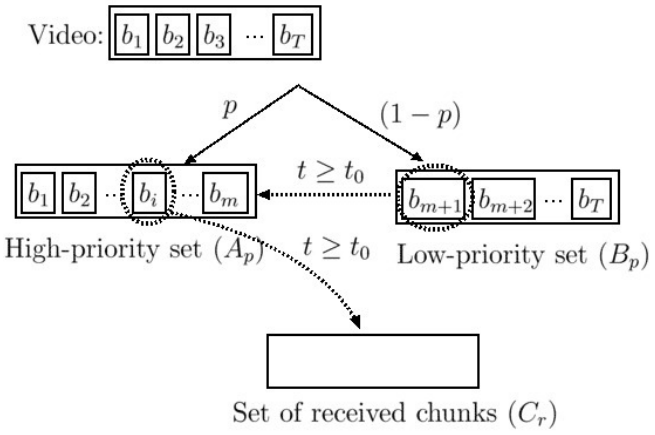
The video chunks to be requested from the neighbor nodes are categorized into three distinct sets: C_r, A_p and B_p. The chunks of set C_r are those that have already been received. The chunks of set A_r are those that have not been received yet and are close in time to be played by the client, i.e., they have a high priority. Set A_r has size of m chunks. At last, the chunks of set B_r are those which have not been retrieved yet and are not close in time to be played by the client, i.e., they have low priority.

A chunk from set A_p is selected with probability p, while a chunk from set B_p is selected with probability $(1 - p)$. Probability p tries to make a compromise between what is needed to be played instantaneously and what is supposed to be played later in the future. This probability may be dynamically adapted in function of the system operation. The chunk-selection policy follows the rarest-first policy.

In case two or more chunks are equally rare, the one that is the closest in time to be played is chosen. After being retrieved, the chunk is inserted in set C_r, from where it will be read by the application and shared among the other nodes. If a chunk is removed from set A_p, then the next chunk of the sequence from set B_p is inserted into A_p. This procedure keeps the size of set A_p at a fixed value (i.e., $|A_p| = m$). In case the chunk is directly retrieved from set B_p, set A_p is not changed.

Figure 4 illustrates the operation of this chunk-selection policy. The video object is divided into T blocks. At $t \leq t_0$, set C_r is empty, i.e., no chunk has been received, and sets A_p and B_p are at their initial states. At $t \geq t_0$, there was the selection of the rarest block from set A_p: chunk b_i. This chunk is then moved to set C_p and the next chunk of the sequence from set B_p (chunk b_{m+1}) is moved to set A_p.

Figure 4: Chunk-selection policy of BiToS and Zhou-Chui-Lui protocols.



$t < t_0$
$A_p = \{b_1, b_2, b_3, ..., b_i, ..., b_m\}$
$B_p = \{b_{m+1}, b_{m+2}, b_{m+3}, ..., b_T\}$
$C_r = \{\}$
$|A_p| = m$
$|B_p| = (T - m)$,
where T is the number of chunks of video and $1 \leq i \leq T$.

$t \geq t_0$
$A_p = \{b_1, ..., b_{i-1}, b_{i+1}, ..., b_m, b_{m+1}\}$
$B_p = \{b_{m+2}, b_{m+3}, ..., b_T\}$
$C_r = \{b_i\}$
$|A_p| = m$
$|B_p| = (T - m - 1)$

The proposal of Zhou-Chiu-Lui [25] is also devoted to a non-interactivity access and is very similar to BiToS protocol. The only difference, comparing to BiToS, is that a greedy policy is used to select a chunk from the high-priority set A_p. That is, the chunks are always retrieved sequentially (in-order).

3.2. Protocol of Shah-Pâris

The protocol of Shah-Pâris [22] is devoted to non-interactive access and is based on the original BitTorrent protocol. However, the chunk-selection policies as well as the neighbor-selection policy are both modified, as detailed next.

Chunk-Selection Policy

The video object is divided into T chunks. A sliding window J_d is defined. It must contain the w chunks (in sequence) that presumably have the highest priority. The chunks belonging to the window J_d are the

only ones that may be requested and this occurs in function of their corresponding rarity: the rarer the chunk is, the earlier it is requested.

The window J_d slides in two situations. In the first situation, it slides when the first chunk is retrieved. It slides as far as its first position corresponds to a chunk that has not been retrieved yet. In the second situation, it slides along an extension of w chunks when a time threshold, denoted as playback delay, expires. Figure 5 depicts the operation of a sliding window of size $w = 5$. We may see that:

Figure 5: Operation of the sliding window of the Shah-Pâris protocol.

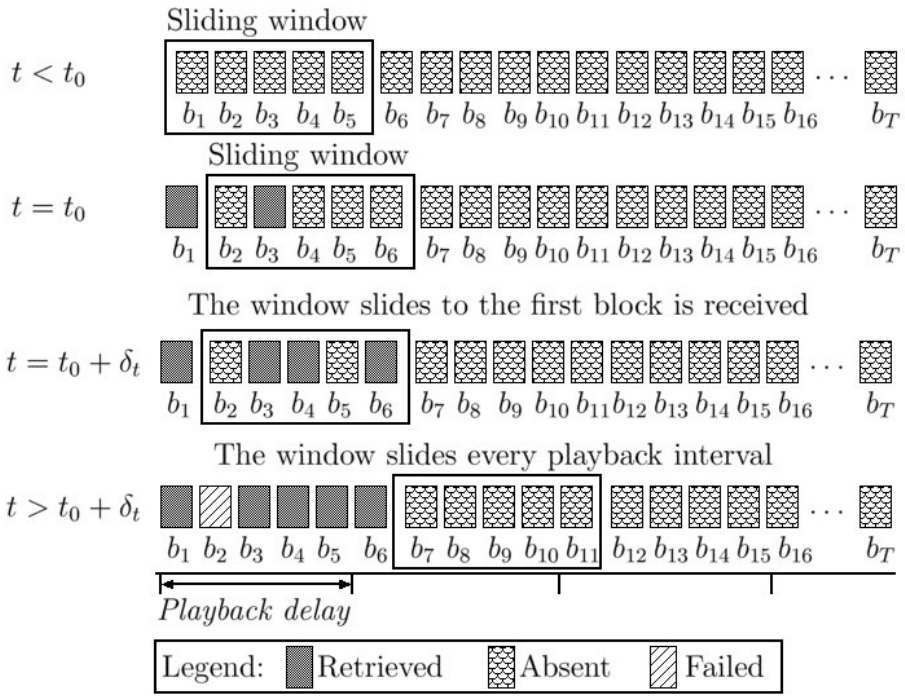
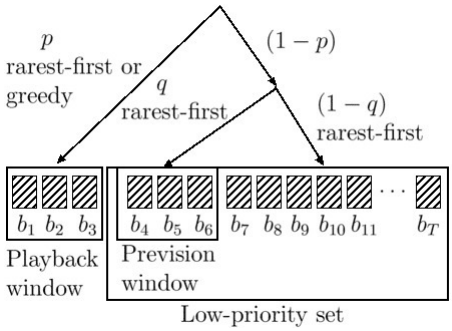


Figure 6: Definition of sets to retrieve chunks from.



see that:

- At $t < t_0$, the window covers chunks b_1, b_2, b_3, b_4, b_5. None of them was retrieved so far.
- At $t = t_0$, chunks b_1 and b_3 were already retrieved and now the window covers chunks b_2, b_3, b_4, b_5, b_6. The first position corresponds to chunk b_2 because this chunk has not been received yet.
- At $t = t_0 + \delta_t$, chunks b_4 and b_6 were already retrieved, but the window has not slide since chunk b_2 has not been retrieved yet.
- At last, at $t > t_0 + \delta_t$, the playback delay has expired. The window must therefore slide w chunks away from its first original position b_1. The first position of the window should

correspond to b_6. However, as chunk b_6 has already been retrieved, the first position corresponds therefore to b_7. In this situation, the window covers chunks b_7, b_8, b_9, b_{10}, b_{11}, and the chunk b_2 may not be retrieved anymore, what may then degrade the quality of visualization.

Neighbor-Selection Policy

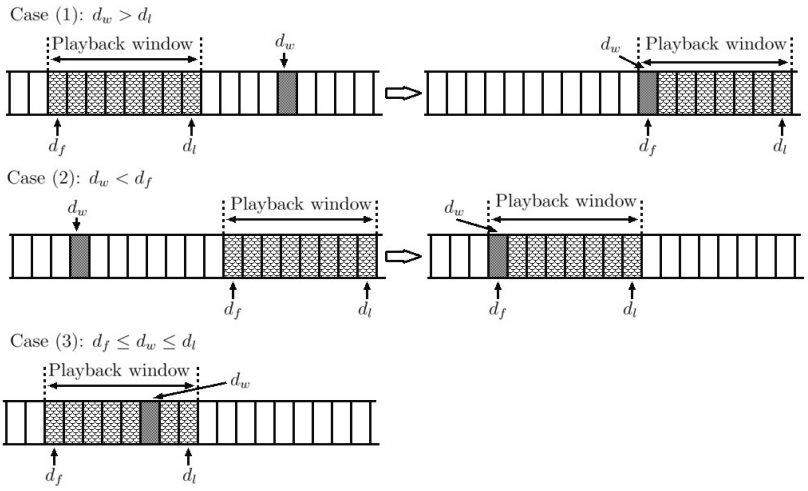
Comparing to the BitTorrent protocol, the only difference lies on the unchoking process. At every window of *w* retrieved chunks, a peer randomly selects *n* neighbors, out of the list sent by the tracker, to be unchoked (see Section 2.2). At all other time instants, the peer follows exactly the same tit-for-tat policy of the BitTorrent protocol.

4. The Novel Protocols

4.1. BitTorrent Interactive Protocol - The First

The BitTorrent Interactive Protocol - The First (BIP-F) is for interactive access, i.e., the client may emulate interactivity actions during the playing of the object. This protocol differs from the BitTorrent protocol exclusively with respect to the chunk-selection policy, as explained next.

Figure 7: Behavior of the playback window when there is a jump.



Firstly, the object chunks are categorized into two sets: *playback window* and B_p. The playback window set contains high-priority chunks, i.e., chunks that are to be soon accessed by the user. This set has *m* consecutive chunks. The B_p set has low-priority chunks, i.e., chunks that will not be soon accessed by the user. This set has size *(T - m)*, where *T* is the total size of the video in chunks. Within this set, it is defined a subset denoted as *prevision window*. This subset contains *k* consecutive chunks, which are determined by the user-behavior model of [7]. Figure 6 depicts this scenario.

Secondly, probabilities are used to select from which set to retrieve chunks. With probability *p*, the chunks are selected from the playback window. With probability *(1 - p)*, the chunks are selected from the B_p set. In case this set is selected, with probability *q* we retrieve chunks from within the prevision window, and, with probability *(1 - q)*, we retrieve chunks from the *prevision window*.

The chunks of the B_p set (including the prevision window) are always selected in function of their rarity. Besides, to select chunks from within the playback window, there are two variants: More-Rare (BIP-FR) and Greedy (BIP-FG). The first variant makes uses of the rarest-first policy to select the chunks, while the second variant makes use of the sequential (greedy) approach.

At last, when there is a sudden change of the current play point, due to an interactivity action (e.g., a jump), the playback window is instantaneously updated. This procedure makes the first position

of the window correspond to the new chunk wanted by the client. The B_p set and the prevision window are updated as well.

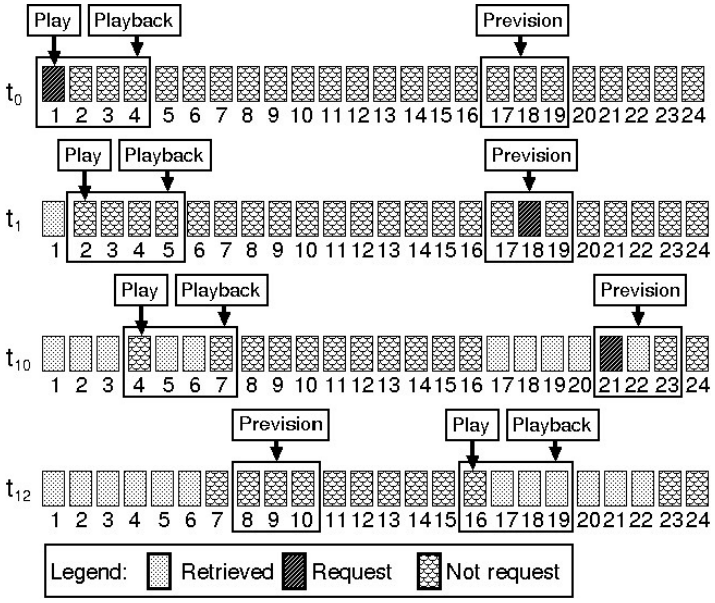
4.2. BitTorrent Interactive Protocol - The Second

The BitTorrent Interactive Protocol - The Second (BIP-S) is also for interactive access. Similarly to BIP-F protocol, this proposal differs from the BitTorrent protocol with respect to the chunk-selection policy, as explained next.

The object chunks are categorized into two sets that are alternately chosen by the peer in order to retrieve the wanted chunks: *playback window* and *prevision window*.

The playback window contains high-priority chunks, i.e., chunks that are to be accessed by the user very soon. This set has m consecutive chunks which are selected in function of their rarity. The prevision window contains k consecutive chunks, which are determined by the user-behavior model of [7] and are retrieved in function of their rarity.

Figure 8: Illustrating the general operations of BIP-S proposal.



When there is a sudden change of the current play point, due to an interactivity action (e.g., a jump), the prevision window is always updated and the playback window may be updated or not, as we detail next. Let d_w be the new first position wanted by the client (after the interactivity action), and let d_f and d_l be the first and last positions, respectively, of the playback window (before the interactivity action). The two following scenarios may occur:

- If $d_w > d_l$ or $d_w < d_f$, then the playback window is updated: it is slide as far as $d_f = d_w$ (Cases 1 and 2 of Figure 7).
- Otherwise, if $d_f \leq d_w \leq d_l$, the playback window does not move (Case 3 of Figure 7).

In Figure 8 we illustrate the general operations of this proposal. We focus on four different instants of time. At time t_0, the user has just arrived in the *swarm*. He immediately starts the data retrieving to play the video as early as possible. The playback window (size of four blocks) initially has blocks 1 (already retrieved), 2, 3 and 4. The prevision window (size of three blocks) has blocks 17, 18 and 19. The play point is block 1, the first block to be played. At time t_1, block 1 is played and the play point becomes the next block. The playback window is moved to the right because block 1 is the first of the window. Then block 18 of the prevision window is solicited because of being the rarest within it.

Now at time t_{10}, the play point is block 4, which has not been retrieved yet. Again the prevision window is chosen and block 21 is solicited. At time t_{12}, the user executes a jump towards block 16. The

play point is then updated and is now within the playback window. The playback window now has blocks 16, 17, 18 and 19. However, note that block 16 has not been retrieved yet and, as early as it is, the window will move entirely to the right as far as the next block to be retrieved. The prevision window is updated, and now covers blocks 8, 9 and 10, which will may possibly be solicited in the near future.

To avoid quality degradation, we have implemented the variant BIP-SB. In this case, there is a buffer to store $x < w$ blocks. The goal of the buffer is to prevent an excessive number of interruptions during the play due to the absence of consecutive chunks. The initial play is only initiated when the buffer is full. If there is a miss (i.e., the wanted block has not been retrieved yet.), the play is stopped and reinitiated when the buffer becomes full again. To clearly state the differences between BIP-S and BIP-F proposals, we outline the points to follow.

- BIP-S retrieves chunks exclusively from the playback and prevision windows.
- BIP-S alternates the chunk retrieving from the playback and prevision windows, i.e., both windows have the same priority.
- In BIP-S protocol, the playback window is updated exclusively when the new play point (after a user's interactive action) is not confined to the current playback window.
- BIP-S has a variant denoted as BIP-SB. This variant makes use of a buffer to prevent degradation due to the absence of consecutive chunks in the client side.
- BIP-F has two variants: BIP-FR and BIP-FG. The first variant makes uses of the rarest-first policy to select the chunks of the playback window, whereas the second variant makes use of a sequential (greedy) approach.

5. Performance Evaluation

5.1. Metrics and Workloads

The metrics considered in the experiments are in Table 1. Two types of workloads are used: real and synthetic. The real workloads refer to the user logs of RIO server [5] captured during the CEDERJ course [5]. A system user may execute the following actions: *play*, *stop*, *pause*, *jump forwards* and *jump backwards*. The synthetic workloads are generated by the user model of [7] and are deployed for behavior prediction. To generate these synthetic workloads, we have used 391 real logs of 20-30 minute sessions.

Both types of workloads are categorized in function of the interactivity level, denoted by the variable *I* and estimated as the average number of requests per session: high interactivity ($15 < I < 40$); medium interactivity ($5 < I < 16$); low interactivity ($0 < I < 6$) and hybrid interactivity ($0 < I < 40$).

Table 1: Metrics.

Metrics	Computation
Mean no. of interruptions (*D*)	$D = \left(\sum_{i=1}^{U} D_i\right)/U$, where D_i is the number of interruptions which took place at user *i* and *U* is the total of users in the swarm.
Mean time to return (*TR*)	$TR = \left(\sum_{i=1}^{U} TR_i\right)/U$, where TR_i is the mean time to return with respect to user *i* and *U* is the total of users in the swarm.
Mean time to begin playing (*TI*)	$TI = \left(\sum_{i=1}^{U} TI_i\right)/U$, where TI_i is the time with respect to user *i* and *U* is the total of users in the swarm.
Download rate (*TxD*)	$TxD = \left(\sum BR_i\right)/\delta_d$, where BR_i is the number of chunks received by the user *i* and δ_d is the time interval between the receiving of the first and the last chunk.
Download Time (*DxT*)	It is defined as the time necessary for a user to download the entire.
Upload rate (*TxU*)	$TxU = \left(\sum BE_i\right)/\delta_u$, where BE_i is the number of chunks sent by the user *i* and δ_u is the time interval between the sending of the first and the last chunks.

To guarantee a significant spectrum of analysis, the workloads are statistically different from each other. The variables used to denote the statistics are in Table 2 and their corresponding values are in Table 3. We may note that the interactivity level of the synthetic workload is 10% to 15% greater or smaller than the real-work interactivity level. The synthetic workload for the hybrid workload shows a greater difference in terms of the interactivity level with respect to the real workload.

5.2. Experiments

The simulation results are obtained by means of the Tangram-II tool [6]. These results are the average of 10 executions, with a confidence interval of 90% varying within 13%, 19%, 31%, 2%, 4% and 2%, respectively, around the average values of the metrics in Table 1. Unless otherwise stated, we consider a single scenario (for all simulations) with the following parameters: object size equal to 1800 s, number of seeds equal to 1, user and seed capacities equal to 100 kB/s.

Table 2: Variable definition.

Variables	Definition
N	Total number of requests.
I	Average number of requests per session (interactivity level).
L	Mean size of segment, measured in seconds.
Std(L)	Standard deviation of *L*.
Coef(L)	Coefficient of variation of *L*.

The values used for the protocol parameters are informed next. These values come from numerous different experiments we have carried out. They may be used for any type of workload, no matter the interactivity level. We chose not to present theses experiments herein for the sake of objectivity. Further details on these values may be found in [8].

- BIP-FR and BIP-FG: q = 0.50; p = 0.80; playback window = 144 chunks; prevision window = L.
- BiToS and Zhou-Chiu-Lui: high-priority set = 144 chunks; p = 0.80.
- Shah-Pâris: sliding window = 144 chunks.
- BIP-S and BIP-SB: playback window = 144 chunks, prevision window = L.
- BIP-SB: buffer = 5 chunks.

To provide an easier understanding and a better organization, we have the experiments separated into five subsections. Subsection 5.2.1 is dedicated to an overall competitive analysis among all protocols herein discussed. The three basic types of workloads (i.e., high, medium and low interactivity) are considered in this analysis. In Subsection 5.2.2, we mainly focus on scalability evaluation. To this end we consider a larger population and the hybrid workload. This time we consider the most competitive protocols coming out from the last subsection.

Table 3: Statistics.

Statistic	Level of Interactivity							
	High		Medium		Low		Hybrid	
	Real	Synthetic	Real	Synthetic	Real	Synthetic	Real	Synthetic
N	1752	1582	1205	1287	388	454	3346	2541
I	24.01	21.68	9.80	10.46	1.99	2.33	8.56	6.50
L	26.03	26.80	61.40	61.70	260.65	260.70	75.54	106.47
Std(L)	29	33	68	65	260	263	77	107
Coef(L)	1.14	1.23	1.11	1.05	1.00	1.00	1.02	1.00

In Subsection 5.2.3, we analyze the BIP-F and BIP-S protocols when the users are erroneously categorized with respect to their interactivity level. The goal of this experimentation is to evaluate how robust the protocol is. In Subsection 5.2.4, we analyze the performance of the protocol of Shah-Pâris when a small buffer is deployed as it happens in the case of the protocol BIP-SB. This experiment has the goal of evaluating the benefits of introducing a buffer in the protocol of Shah-Pâris and, more fairly, compare it with our proposals. At last, Subsection 5.2.5 evaluates the fairness of the BIP-S and Shah-Pâris protocols.

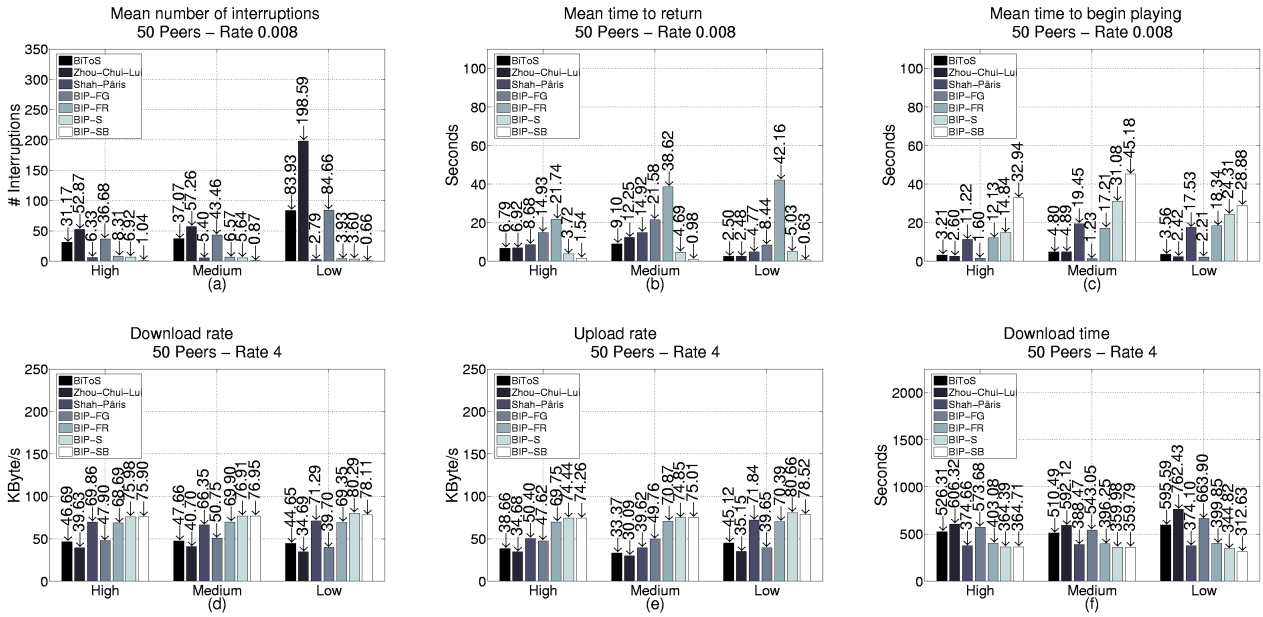
5.2.1. Competitive Analysis

In this experiment, the proposals are evaluated in function of the user's interactivity level. To this end, we use workloads of three distinct levels: high, medium and low. Throughout this section we set the user's arrival rate to the swarm to be $\lambda = 4$ and 0.008 *users/sec*. Nonetheless, the truth is that the experiments do not reveal any significant difference whether we consider $\lambda = 4$ or 0.008. That means that there is no difference between a burst of users or users arriving more regularly spaced. In the following, we separately discuss the main results we obtain in the experiments.

Mean number of interruptions (Figure 9(a)): the proposal of Zhou-Chiu-Lui presents the greatest number of interruptions, followed by the proposals BiToS and BIP-FG, no matter the interactivity level and user's arrival rate. As for the proposals of Shah-Pâris, BIP-FR, BIP-S and BIP-SB, which deploy a rarest-first retrieving mechanism, we may notice the following: the higher the interactivity level is, the greater the number of interruptions is. A partial conclusion is thus that protocols that make use of a sequential retrieving mechanism tend to have a greater number of interruptions, while those that deploy a rarest-first mechanism tend to have less interruption.

Mean time to return (Figure 9(b)): the protocols that deploy a sequential-retrieving mechanism have a better performance. This is already expected since the time to recover from a block miss (i.e., when the block to be played has not been retrieved yet) tends to be shorter. On the other hand, the protocols that deploy a rarest-first mechanism tend to recover more slowly. We point out that the protocol BIP-FR has the greatest mean time. We still outline that the good performance of the protocols BIP-S and BIP-SB are basically due to the deployment of the predicting-behavior model which reduce the number of interruptions (Figure 9(a)) and, consequently, the mean time to return (Figure 9(b)).

Figure 9: Competitive analysis among all proposals, for λ = 4 and 0.008 users/second.



Mean time to begin playing (Figure 9(c)): the protocols which make use of a sequential-retrieving mechanism are more competitive. This is already expected since this mechanism will always retrieve the first block that is to be played. We outline that the protocol BIP-FG is the one of best performance of all, while the Protocol BIP-SB is the worst. The poor performance of the protocol BIP-SB is due to the time that is necessary to fill out the buffer.

Download rate, Upload rate, Download time (Figures 9(d), 9(e) and 9(f), respectively): as expected, the protocols which make use of the rarest-first retrieving mechanism have a superior performance than those which do not. We outline that the protocols BIP-S and BIP-SB have the best performance.

In brief, we may thus conclude that: (i) the deployment of a sequential-retrieving mechanism is not recommended since the rarest-first mechanism is usually more efficient; (ii) the deployment of three or more sets of blocks in order to settle priorities introduces an additional complexity to the protocol and does not turn out to be efficient; (iii) the deployment of a predicting-behavior model may be helpful mainly for the optimization of the metric mean time to return; (iv): the most competitive proposals are the one of Shah-Pâris, BIP-FR, BIP-FG, BIP-S and BIP-SB.

5.2.2. Scalability

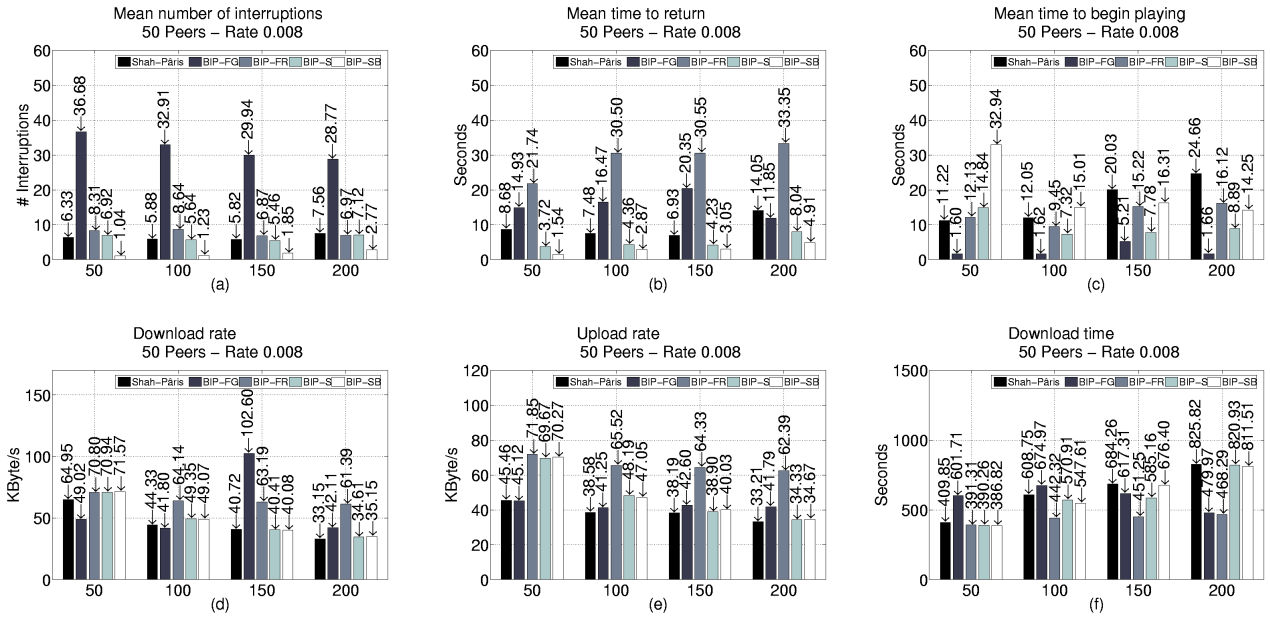
In this section we try to evaluate how the protocols are affected due to a variation in the number of peers in the swarm. To this end, we consider a swarm of 50, 100, 150 and 200 peers. We outline that we herein use the hybrid workload and the most competitive protocols coming out from the last subsection, namely: of Shah-Pâris, BIP-FR, BIP-FG, BIP-S and BIP-SB. As before, we discuss the results separately, considering each of the metrics individually.

Mean number of interruptions (Figure 10(a)): the sequential-retrieving mechanism does not have a satisfactory performance. On the other hand, the rarest-first mechanism is able to provide relevant optimizations. For example, the protocol BIP-SB has the best performance of all, even though there is an increase at the number of interruptions. This result mainly reinforces the importance of the rarest-first mechanism, the deployment of a buffer as well as the deployment of a predicting-behavior model.

Mean time to return (Figure 10(b)): The protocols BIP-S and BIP-SB present the best performances. This is mainly due to the deployment of the predicting-behavior model that helps to reduce the number of interruptions as well as to optimize the time needed to recover from a block miss. On the other hand, the protocols BIP-FG and BIP-FR do not have a satisfactory performance. Even though they make use of a predicting-behavior model, they deploy three distinct sets from where to retrieve blocks. This introduces complexity and negatively impacts on performance. The protocol of Shah-Pâris does not have a satisfactory performance either. This is mainly because of the non-deployment of a predicting-behavior model.

Mean time to begin playing (Figure 10(c)): the protocol BIP-FG shows a very competitive performance. This result outlines the strength of the sequential-retrieving mechanism for optimizing this metric. The other protocols that make use of a rarest-first mechanism have an inferior performance, as already expected. Amazingly, the protocol of Shah-Pâris presents a significant increase at this metric in function of the number of peers in the swarm, while the protocol BIP-SB has a decrease at this same metric in function of the number of peers. This is due to the deployment of more than a single set from where to retrieve blocks, what may increase the number and availability of blocks in the swarm.

Figure 10: Competitive evaluation of the protocols Shah-Pâris, BIP-S and BIP-SB for a hybrid workload and λ = 0.008 and 4 users/second.



Download rate, Upload rate, Download time (Figures 10(d), 10(e) and 10(f), respectively): the protocols have quite similar performances. They all have reductions at the rates and an increase at the download time as the number of users in the swarm increases. This is partly explained by the fact that we have only one seed in the swarm, thus creating a bottleneck for distributing the blocks.

From the experiments, we may thus briefly state that: (i) the protocols that make use of the rarest-first retrieving mechanism are more efficient than the ones the deploy the sequential paradigm; (ii) the protocols BIP-S and BIP-SB outperform the other protocols herein investigated; (iii) the deployment of a predicting-behavior model reveals to be quite important for interactive VoD distribution.

5.2.3. Robustness

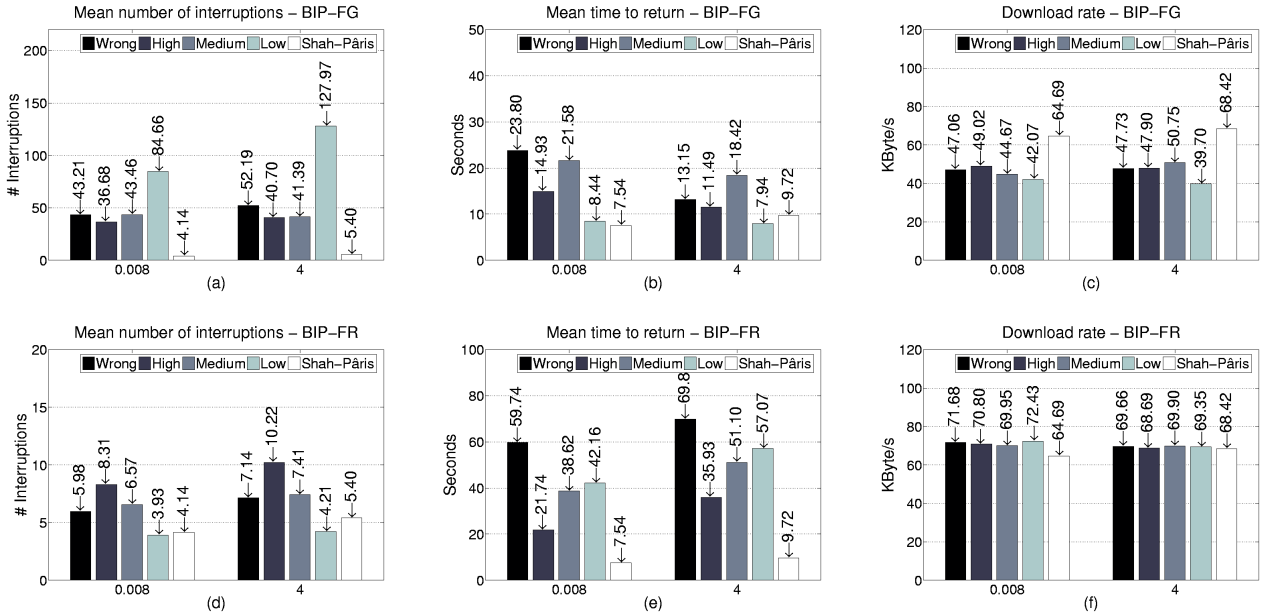
The main optimizations achieved by our proposals are especially due to the deployment of the user-behavior prediction model. We though wonder what the impact is when this prediction fails. To this end, we consider a set of 50 users of high and low interactivity. For each of these users, it is generated a behavior prediction of a different level of interactivity. More precisely, a high-interactivity user receives a low-interactivity prediction and vice versa. For ease of reference, the results for these experiments are shown under the label of *wrong workload* in the pictures to appear.

We mention that we compare all results of this section with those that were generated considering a correct assignment of the user-interactivity level. Besides, we also consider the Shah-Pâris proposal in order to see if it outperforms our protocols (BIP-F and BIP-S) when a wrong assignment is made. The metrics *mean time to return*, *upload rate* and *download time* do not vary significantly and thus are not herein discussed. As before, we discuss the results separately.

BIP-FG and BIP-FR Proposals

Mean number of interruptions (Figure 11(a)): The BIP-FG proposal presents a similar performance. This same observation is true for the metric *mean time to return* (Figure 11(b)) and the metric download rate (Figure 11(c)) as well.

Figure 11: Performance evaluation in the presence of a wrong assignment of user interactivity.



Mean number of interruptions (Figure 11(d)): no significant variation is observed in the BIP-FR proposal. For the metric *download rate* (Figure 11(f)), no variation is noticed either. However, for the metric *mean time to return* (Figure 11(e)), it becomes clear that there is an inferior performance.

BIP-FG, BIP-FR and Shah-Pâris

Comparing the BIP-FR and the protocol of Shah-Pâris, we have quite similar performances considering the metrics *mean number of interruptions* and *download rate*. As for the *mean time to return*, we have that the proposal of Shah-Pâris provides more optimized outcomes.

Now comparing BIP-FG and the protocol of Shah-Pâris, we again see that the chunk sequential-retrieving mechanism does not show a good performance since for all competitive metrics the proposal of Shah-Pâris is superior.

BIP-S and BIP-SB Proposals

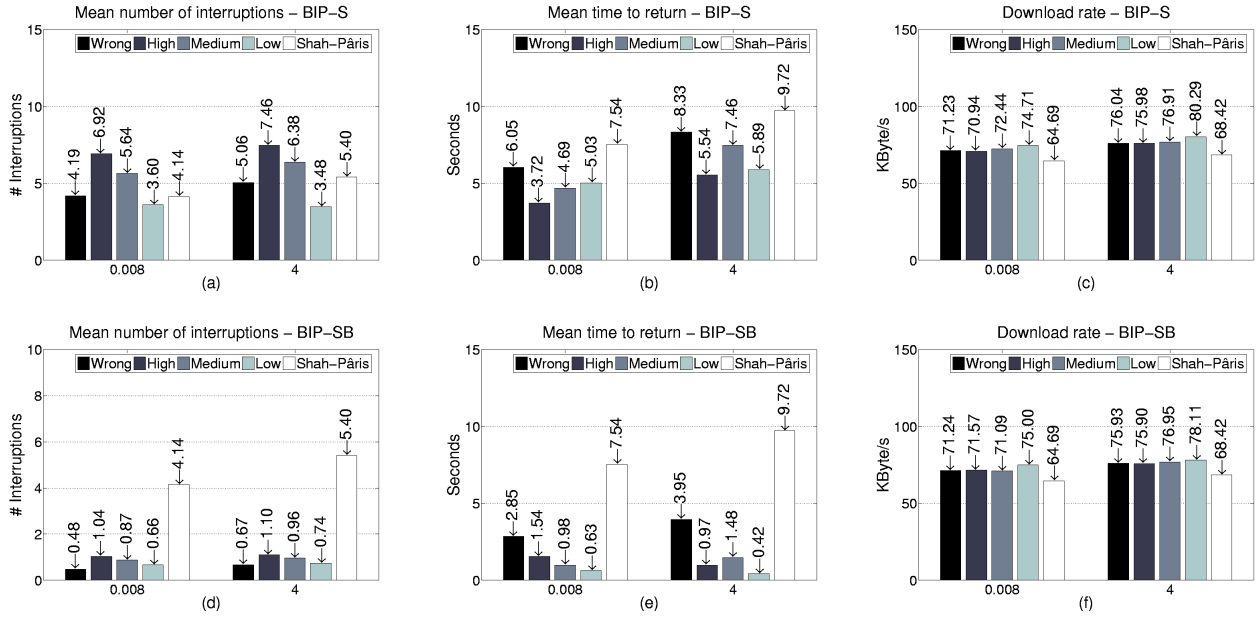
Mean time to return (Figures 12(b) and 12(d)): this result shows to be 45% greater than that considering a correct prediction. This surely confirms the importance of assigning a correct categorization for the interactivity level.

Mean number of interruptions (Figures 12(a) and 12(e)): the results obtained for high-interactivity users as well as for low-interactivity users are not that different from those that are obtained when a correct categorization of the interactivity level is assigned. We may thus conclude that there is no impact on this metric. This same observation applies to the metric *download rate* (Figures 12(c) and 12(f)).

BIP-S, BIP-SB and Shah-Pâris

Focusing on the BIP-S and the proposal of Shah-Pâris, we obtain the following observations. For the *mean number of interruptions* (Figure 12(a)), we have that there is a quite similar performance; for the *mean time to return* (Figure 12(b)), the BIP-S protocol presents a value that is 20% smaller than that of Shah-Pâris; for the download rate (Figure 12(c)), the BIP-S protocol shows values that are usually 20% greater than that of Shah-Pâris.

Figure 12: Performance evaluation in the presence of a wrong assignment of user interactivity.



Now when we compare the BIP-SB protocol with the proposal of Shah-Pâris we have that the former is far more efficient than the latter. The final results indicate differences that reach up to 90%. Please refer to Figure 12(d) (*mean number of interruptions*), Figure 12(e) (*mean time to return*) and Figure 12(f) (*download rate*).

Finally, judging from all above experiments, we may thus state that: (i) the greatest impact due to a wrong interactivity-level assignment happen to the metric *mean time to return*. This is already expected since this is the one that is mostly favored by the behavior-predicting model; (ii) the BIP protocol outperforms the proposal of Shah-Pâris even when there is a wrong assignment of the user-interactivity level.

5.2.4. Buffer Deployment

From the results so far observed and not taking account our proposals, we may state that the protocol of Shah-Pâris is the most efficient one. We then wonder if this protocol may be improved by deploying a buffer, as it happens to the BIP-SB protocol. The goal of this section is thus to carry out this evaluation.

We consider 50 high-interactivity peers and a single seed. The arrival rate is sometimes $\lambda = 4$ and sometimes $\lambda = 0.008$ users/second. The truth though is that the experiments do not reveal any significant difference whether we consider $\lambda = 4$ or $\lambda = 0.008$. As before, that means that there is no difference between a burst of users or when users arrive more regularly spaced. We also consider that the buffer size corresponds to 5, 15 or 20 blocks. In the following, we separately discuss the main results we obtain for each of the competitive metrics.

Mean number of interruptions (Figure 13(a)): there is no significant difference among the proposals, no matter the buffer size. This means that the protocol of Shah-Pâris has been improved.

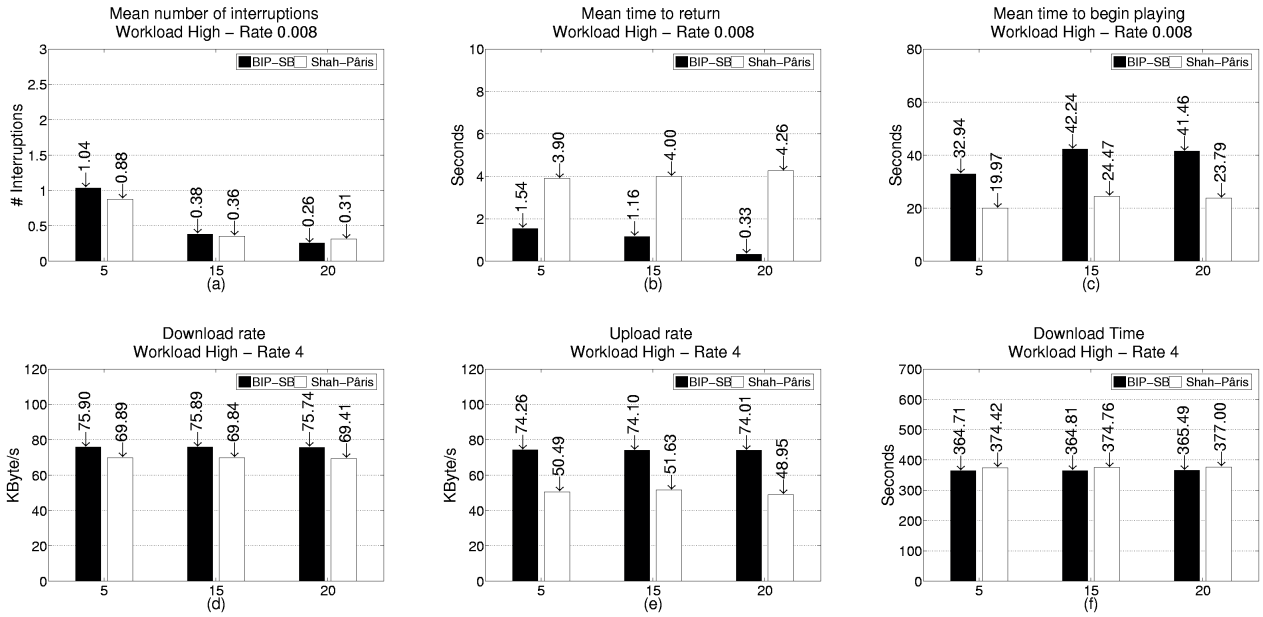
Mean time to return (Figure 13(b)): the protocol BIP-SB presents a reduction of 96% with respect to the value obtained by the protocol of Shah-Pâris with buffer. This is explained by the deployment of the predicting-behavior model.

Mean time to begin playing (Figure 13(c)): the protocol of Shah-Pâris shows to be more efficient than the protocol BIP-SB. This better behavior may be explained by the sequential-retrieving mechanism which always favors the optimization of this metric.

Download rate, Upload rate, Download time (Figures 13(d), 13(e) and 13(f), respectively): the proposal of BIP-SB presents an optimization of 12%, 50% and 10%, respectively, comparing to the

proposal of Shah-Pâris with buffer. This superior performance may be explained due to the deployment of two distinct sets from where to retrieve blocks, what leads to a better distribution of blocks among the nodes.

Figure 13: Performance evaluation of Shah-Pâris when using a buffer with λ= 0.008 and 4 users/second.



In brief, we may conclude that: (i) the deployment of a buffer may help to reduce the number of interruptions and consequently the mean time to return; (ii) the deployment of a buffer may also help to keep high rates of upload and download; (iii) even though the protocol of Shah-Pâris has considerably been improved, the protocol BIP-SB is still more efficient.

5.2.5. Fairness

To evaluate the fairness of the proposals, we consider these three statistics: the mean ($\bar{x}$); the variance (σ^2); and the difference between the maximum and the minimum value (d). These statistics are calculated for the following metrics: *mean time to return* and *download rate*. The first is intended to evaluate the playback quality and the second the performance of a node with respect to others.

We mention that we chose not to consider the BIP-FG and BIP-FR protocols in this evaluation. The motivation for that was because the results obtained in the last sections indicated that these protocols are not as efficient as the BIP-S and BIP-SB protocols.

Mean time to return (Table 4): the proposal of Shah-Pâris presents a greater variance (σ^2) and difference (d). This reveals a low level of fairness. On the other hand, the protocols BIP-S and BIP-SB present smaller values for the variance (σ^2) and difference (d). This means that these protocols are fairer than that of Shah-Pâris. This higher level of fairness possibly comes from the deployment of the predicting-behavior model, which positively impacts on the overall performance of these protocols since it makes the values stable.

Download time (Table 4): the protocol of Shah-Pâris presents high values for the variance (σ^2) and difference (d). This is possibly due to the intrinsically interactive behavior of the users. These users often jump from block to block, thus negatively impacting on the overall performance, mainly because of the interruptions. This is not totally perceived by the protocols BIP-S and BIP-SB, since they deploy a predicting-behavior model.

Table 4: Fairness.

Workload	Protocol	Mean Time to Return (s)			Download Rate (kB/s)		
		$\bar{x}$	σ^2	d	$\bar{x}$	σ^2	d
High	Shah-Pâris	8.98	224.73	77.35	66.24	115.14	41.75
	BIP-S	3.89	8.55	13.38	71.41	3.70	11.00
	BIP-SB	3.19	11.84	13.29	71.66	2.51	8.04
Medium	Shah-Pâris	14.91	447.24	91.84	64.12	180.66	45.04
	BIP-S	4.8	14.53	17.04	73.89	0.30	1.98
	BIP-SB	0.98	2.81	7.89	73.57	0.40	2.32
Low	Shah-Pâris	4.77	28.50	22.68	67.19	74.68	41.45
	BIP-S	5.03	8.18	13.43	76.20	0.009	0.45
	BIP-SB	0.62	1.29	5.98	76.50	0.009	0.51

From the experiments, we may thus briefly state that: (i) the protocol BIP-SB is the fairest one; (ii) the fairness of the protocols BIP-S and BIP-SB are mainly determined by the deployment of a predicting-behavior model; and (iii) the protocol of Shah-Pâris is the unfairness one.

6. Conclusions and Future Work

This work presented two novel protocols for interactive video-on-demand service: the BitTorrent Interactive Protocol - The first (BIP-F) and the BitTorrent Interactive protocol - The second (BIP-S). Both proposals are based on the well-known BitTorrent Protocol and are designed for object multimedia delivery in mesh architectures. From a user experience point-of-view, the main goal of this work was to contribute for a solution with low latency to begin playback as well as jitter-free playback.

The protocols BIP-F and BIP-S basically define chunk sets and deploy a user-behavior predicting model to possibly forecast the multimedia data to be played by the user. These sets are dynamically updated in function of the client's interactive actions during the play. The same politics of the original BitTorrent protocol is used to select nodes from whom the chunks are retrieved. The validation of these proposals is done through simulations using workloads generated from real user-logs of a multimedia server (RIO - Random I/O System) [5]. Distinct performance metrics are considered and competitive analyses are carried out considering other proposals of the literature. Among the most important results we obtained in the experiments, we may outline these two following ones:

- Comparing to other protocols of the literature, our proposals presented an average optimization of up to: (i) 94% at the metric mean number of interruptions and 90% at the metric mean time to return. These results were rather influenced by the behavior-predicting model deployed by these protocols; (ii) 82% at the metric mean time to begin playing. This metric is influenced by the chunk selection algorithm which, in this case, is based on a sequential retrieving mechanism; (iii) 40% at the metrics download and uploads rates.
- A wrong assignment of the user-interactivity level implies increases of up to 83% at the metric mean time to return. No significant changes were though observed at the other metrics. In spite of that, our protocols were still more efficient than the others of the literature.

We believe future work may include studying our proposals in a more heterogeneous environment. For instance, we may consider peers of different bandwidth capacities and more than one seed to distribute the objects. We also consider evaluating a variant of the BIP-S protocol that implements a buffer together with a chunk sequential-retrieving mechanism. At last, we may look into different neighbor-selection policies to see if there is still space for possible improvements.

References

[1] Borghol, Y., Ardon, S., Carlsson, N., Mahanti, A. (2010). Toward efficient on-demand streaming with bittorrent. In: Proc. IFIP Networking, Chennai, India.

[2] Carlsson, N., Eager, D. L. (2007). Peer-assisted On-demand Streaming of Stored Media using BitTorrent-like Protocols. In: IFIP/TC6 Networking, Atlanta, GA, USA, 570–581.

[3] Chu, Y., Rao, S. G., Zhang, H. (2000). A Case for End System Multicast. In: ACM SIGMETRICS international conference on Measurement and modeling of computer systems, Santa Clara, California, USA, 1–12.

[4] Cohen, B. (2003). Incentives Build Robustness in BitTorrent. In: First Workshop on Economics of Peer-to-Peer Systems, Berkeley, CA, USA.

[5] De Souza e Silva, E., Leão, R. M. M., Santo, A. D., Azevedo, J. A., Netto, B. C. M. (2006). Multimedia Supporting Tools for the CEDERJ - Distance Learning Initiative applied to the Computer Systems Course. In: 22th ICDE World Conference on Distance Education, Rio de Janeiro, RJ, Brasil, 1–11.

[6] De Souza e Silva, E., Figueiredo, D., Leão, R. M. M. (2009). The TANGRAM-II Integrated Modeling Environment for Computer Systems and Networks. In: ACM SIGMETRICS Performance Evaluation Review 36 (4), 45–65.

[7] De Vielmond, C. C. L. B., Leão, R. M. M., de Souza e Silva, E. (2007). Um modelo HMM hierárquico para usuários interativos acessando um servidor multimedia. In: Simpósio Brasileiro de Redes de Computadores, Vol. I, Belém, Pará, Brasil. In Portuguese.

[8] Filho, L. J. H., Rodrigues, C. K. S., Leão, R. M. M. (2009). Acesso interativo para aplicações P2P de streaming de vídeo. In: XVII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos, Recife, PE, Brasil, 599–612. In Portuguese.

[9] Garbacki, P., Epema, D. H. J., Pouwelse, J. (2008). Offloading servers with collaborative video on demand. In: Proc. IPTPS, Tampa Bay, FL, USA.

[10] Guo, Y., Suh, K., Kurose, J., Towsley, D. (2007). P2Cast: peer-to-peer patching for video-on-demand service. In: Multimedia Tools and Applications, 33 (2) 109–129.

[11] Guo, Y., Mathur, S., Ramaswamy, K., Yu, B., Shengchao, P. (2007). PONDER: Performance Aware P2P Video-on-Demand Service. In: IEEE Global Telecommunications Conference - GLOBECOM, Washington, DC, USA, 225–230.

[12] Guo, Y., Suh, K., Kurose, J., Towsley, D. (2008). DirectStream: A directory-based peer-to-peer video streaming service. In: Journal of Computer communications - COMCOM 31, 520–536.

[13] Huang, C., Li, J., Ross, K. W. (2007). Can internet video-on-demand be profitable?. In: SIGCOMM Comput. Commun. Rev. 37 (4) 133–144.

[14] Huang, C., Li, J., Ross, K. (2007-2). Peer-assisted VoD: Making internet video distribution cheap. In: 6th International Workshop on Peer-to-Peer Systems (IPTPS'07), Bellevue, WA.

[15] Huang, Y., Fu, T. Z., Chiu, D.-M., Lui, C., Huang, John C.S. (2008). Challenges, design and analysis of a large-scale P2P VoD system. In: ACM SIGCOMM Conference on Data Communication, Seattle, WA, USA, 375–388.

[16] Jannotti, J., Gifford, D. K., Johnson, K. L., Kaashoek, M. F., O'Toole Jr, J.W. (2000). Overcast: reliable multicasting with on overlay network. In: 4th Conference on Symposium on Operating System Design & Implementation, San Diego, CA, USA, 14–20.

[17] Legout, A., Urvoy-Keller, G., Michiardi, P. (2006). Rarest First and Choke Algorithms are enough. In: 6th ACM SIGCOM - Conference on Internet Measurement, Rio de Janeiro, RJ, Brazil, 203–216.

[18] Li, B., Xie, S., Keung, G., Liu, J., Stoica, I., Zhang, H., Zhang, X., (2007). An Empirical Study of the Coolstreaming+ System. In: IEEE Journal on Selected Areas in Communications 25 (9), 1627 – 1639.

[19] Rodrigues, C. K. S., Leão, R. M. M. (2007). Bandwidth usage distribution of multimedia servers using Patching. In: Computer Networks 51, 569–587.

[20] Rodrigues, C. K. S., Filho, L. J. H., Leão, R. M. M. (2008). On Scalable Interactive Video-On-Demand Services. In: European Journal of Scientific Research 21 (4), 662–686.

[21] Savolainen, P., Raatikainen, N., Tarkome, S. (2008). Windowing BitTorrent for video-on-demand: Not all is lost with tit-for-tat. In: Proc. IEEE GLOBECOM, New Orleans, LA, USA.

[22] Shah, P., Pâris, J.-F. (2007). Peer-to-Peer Multimedia Streaming Using BitTorrent. In: IEEE International Performance, Computing, and Communications Conference - IPCCC, New Orleans, Louisiana, USA, 340–347.

[23] Vlavianos, A., Iliofotou, M., Faloutsos, M. (2006). BiToS: Enhancing BitTorrent for Supporting Streaming Applications. In: 9th IEEE Global Internet Symposium, Barcelona, Spain.

[24] Wang, F., Xiong, Y., Liu, J. (2007). mTreebone: A Hybrid Tree/Mesh Overlay for Application-Layer Live Video Multicast, in: 27th International Conference on Distributed Computing Systems - ICDCS, Toronto, Ontario, Canada.

[25] Zhou, Y., Chui, D. M., Lui, J. C. S. (2007). A Simple Model for Analyzing P2P Streaming Protocols. In: IEEE International Conference on Network Protocols - ICNP, Beijing, China, 226–235.