International Journal of Digital Multimedia Broadcasting

# Cloud Computing and Dynamic Resource Allocation for Multimedia Applications

Guest Editors: Yifeng He, Ling Guan, Wenwu Zhu, and Ivan Lee

# Cloud Computing and Dynamic Resource Allocation for Multimedia Applications

International Journal of
Digital Multimedia Broadcasting

# Cloud Computing and Dynamic Resource Allocation for Multimedia Applications

Guest Editors: Yifeng He, Ling Guan, Wenwu Zhu, and Ivan Lee

# Editorial Board

# Contents

Hindawi Publishing Corporation
International Journal of Digital Multimedia Broadcasting
Volume 2012, Article ID 238460, 2 pages
doi:10.1155/2012/238460

*Editorial*

# Cloud Computing and Dynamic Resource Allocation for Multimedia Applications

**Yifeng He,[1] Ling Guan,[1] Wenwu Zhu,[2] and Ivan Lee[3]**

[1] *Department of Electrical and Computer Engineering, Ryerson University, Toronto, ON, Canada M5B 2K3*
[2] *Department of Computer Science, Tsinghua University, Beijing 100084, China*
[3] *School of Computer and Information Science, University of South Australia, Adelaide, SA 5095, Australia*

Correspondence should be addressed to Yifeng He, yhe@ee.ryerson.ca

Received 22 December 2011; Accepted 22 December 2011



We have witnessed significant advances in multimedia applications due to the rapid increase in digital media, computing power, communication speed, and storage capacity. Multimedia has become an indispensable aspect in contemporary daily life. Recently, cloud computing technology has offered great opportunities for multimedia applications. The performance of the multimedia applications can be significantly improved by optimizing the various resources in the multimedia system. The papers selected for this special issue represent a good panel for addressing the resource allocation problems in multimedia applications. We would like to thank the authors for their excellent contributions. We are grateful to all reviewers for their constructive comments which help to improve the quality of the papers.

This special issue contains five papers, in which two papers are related to cloud-based multimedia applications, two papers deal with Peer-to-Peer (P2P) video streaming systems, and one paper addresses the resource allocation problem in wireless networks.

In the paper entitled "*Multi-objective genetic algorithm for task assignment on heterogeneous nodes*," C. B. P. Del Notario et al. present a task assignment strategy based on genetic algorithms for a client-cloud multimedia system. Specifically, the task execution quality is maximized under the constraints of energy and bandwidth. In the video processing scenario, the proposed method employs trans-coding to provide a tradeoff between video quality and processing and bandwidth demands.

In the paper entitled "*Towards an automatic parameter-tuning framework for cost optimization on video encoding cloud*," X. Li et al. conduct an empirical study on video encoding cloud, targeting an optimization framework to minimize H.264 encoding cost by tuning the key parameters. The experiment results show that the tested parameters can be independently tuned to minimize the encoding cost, which makes the automatic parameter-tuning framework feasible and promising for video encoding cloud.

In the paper entitled "*Improving streaming capacity in multi-channel P2P VoD systems via intra-channel and cross-channel resource allocation*," Y. He and L. Guan. present novel methods to improve the streaming capacity for a multi-channel P2P video-on-demand (VoD) system by efficiently utilizing both intrachannel and cross-channel resources. The intrachannel resource allocation problem can be formulated into a linear programming (LP) problem. The cross-channel resource allocation can be enabled by both optimal server upload allocation among channels and cross-channel sharing of peer upload bandwidths.

In the paper entitled "*Performance evaluation of an object management policy approach for P2P networks*," D. Vieira et al. present an object management approach for video web cache in a P2P network, taking advantage of object's metadata, for example, video popularity, and object's encoding techniques, for example, scalable video coding (SVC). In addition, the paper examines the impact of churn on the performance of object management policies.

In the paper entitled "*A gain-computation enhancements resource allocation for heterogeneous service flows in IEEE 802.16 m mobile networks*," W. B. Hassen and M. Afif. examine resource allocation problems for orthogonal-frequency-division-multiple-access-(OFDMA-) based wireless mobile networks. In the single-service case, a dynamic resource

allocation algorithm is proposed to maximize the system capacity while ensuring fairness among users. In the multiservice case, an adaptive resource allocation algorithm is proposed to support quality-of-Service-(QoS-) sensitive applications such as streaming media.

*Yifeng He*
*Ling Guan*
*Wenwu Zhu*
*Ivan Lee*

Hindawi Publishing Corporation
International Journal of Digital Multimedia Broadcasting
Volume 2012, Article ID 807520, 9 pages
doi:10.1155/2012/807520

*Research Article*

# Improving Streaming Capacity in Multi-Channel P2P VoD Systems via Intra-Channel and Cross-Channel Resource Allocation

**Yifeng He and Ling Guan**

*Department of Electrical and Computer Engineering, Ryerson University, Toronto, ON, Canada M5B2K3*

Correspondence should be addressed to Yifeng He, yhe@ee.ryerson.ca

Received 13 August 2011; Accepted 6 December 2011

Academic Editor: Ivan Lee



Multi-channel Peer-to-Peer (P2P) Video-on-Demand (VoD) systems can be categorized into *independent-channel* P2P VoD systems and *correlated-channel* P2P VoD systems. Streaming capacity for a channel is defined as the maximal streaming rate that can be received by every user of the channel. In this paper, we study the streaming capacity problem in multi-channel P2P VoD systems. In an independent-channel P2P VoD system, there is no resource correlation among channels. Therefore, we can find the average streaming capacity for the independent-channel P2P VoD system by finding the streaming capacity for each individual channel, respectively. We propose a distributed algorithm to solve the streaming capacity problem for a single channel in an independent-channel P2P VoD system. The average streaming capacity for a correlated-channel P2P VoD system depends on both the intra-channel and cross-channel resource allocation. To better utilize the cross-channel resources, we first optimize the server upload allocation among channels to maximize the average streaming capacity and then propose cross-channel helpers to enable cross-channel sharing of peer upload bandwidths. We demonstrate in the simulations that the correlated-channel P2P VoD systems with both intra-channel and cross-channel resource allocation can obtain a higher average streaming capacity compared to the independent-channel P2P VoD systems with only intra-channel resource allocation.

## 1. Introduction

Video-on-demand (VoD) services have been attracting a lot of users because it allows users to watch any video at any time. Traditional client/server architectures for VoD services cannot provide video streams to a large number of concurrent users. To offload the server upload burden, Peer-to-Peer (P2P) technology has been integrated into VoD applications by utilizing the uplink bandwidths of the peers [1–5].

Most of the P2P VoD systems offer many video channels. Users can choose any of the channels that they are interested in at any time. The P2P VoD systems with multiple channels are called *multi-channel* P2P VoD systems. Depending on the resource correlation, multi-channel P2P VoD systems can be categorized into *independent-channel* P2P VoD systems and *correlated-channel* P2P VoD systems. In an independent-channel P2P VoD system, the peers watch the same channel form an independent overlay and share the resources with each other exclusively within the overlay. In a correlated-channel P2P VoD system, overlay $m$ formed by the peers watching channel $m$ can be correlated with overlay $k$ formed by the peers watching channel $k$, such that the peers in overlay $m$ can share the resources not only with other peers in overlay $m$ but also with the peers in overlay $k$. The resources in a correlated-channel P2P VoD system can be utilized in a better way compared to an independent-channel P2P VoD system.

In P2P VoD systems, users would like to watch the video at a high quality. The streaming rate can be used to indicate the video quality. Let $\mathbf{M}$ denote the set of the channels in a P2P VoD system. If channel $m (m \in \mathbf{M})$ is associated with overlay $m$, *streaming capacity* for channel $m$, denoted by $c_m$, is defined as the maximum streaming rate that can be received by every user in overlay $m$ [6, 7]. The *average streaming capacity* $c_{\text{avg}}$ for a multi-channel P2P VoD system is defined as $c_{\text{avg}} = \sum_{m \in \mathbf{M}} p_m c_m$ where $p_m$ is the priority of channel $m$,

and $\Sigma_{m\in\mathbf{M}} p_m = 1$. The priority of a channel can be set by the service provider. For example, the more expensive channel can be assigned a higher priority.

Streaming capacity problem in multi-channel P2P VoD systems is a challenging problem. The average streaming capacity in a multi-channel P2P VoD system is dependent on the number of the peers, the playback time, and the bandwidth of each peer, the server capacity, the overlay construction, and the resource allocation. Optimal resource allocation in a multi-channel P2P VoD system is expected to improve the streaming capacity. However, resource allocation in multi-channel P2P VoD systems is quite challenging due to the following reasons. (1) Peers have heterogeneous upload and download bandwidths and different playback progress. (2) Each channel is heterogeneous in terms of available resources, since the number of the peers in each channel is different. (3) Peers may leave or join a channel dynamically.

In this paper, we improve the average streaming capacity for multi-channel P2P VoD systems by better utilizing both the intra-channel resources and the cross-channel resources. We first investigate the streaming capacity for an independent-channel P2P VoD system, in which we find the streaming capacity for a single channel by optimizing the intra-channel resource allocation in a distributed manner. We then investigate the streaming capacity for a correlated-channel P2P VoD system, in which we find the average streaming capacity for multiple channels by optimizing both intra-channel and cross-channel resource allocation. The proposed cross-channel resource allocation consists of two steps as follows. (1) We optimize the server upload allocation among channels to maximize the average streaming capacity. (2) We introduce *cross-channel helpers* to establish cross-channel links and then utilize cross-channel peer upload bandwidths to improve the average streaming capacity.

The remainder of this paper is organized as follows. Section 2 discusses the related work. Section 3 formulates and solves the streaming capacity problem for an independent-channel P2P VoD System. Section 4 finds the average streaming capacity for a correlated-channel P2P VoD System. The simulation results are provided in Section 5, and the conclusions are drawn in Section 6.

## 2. Related Work

Streaming capacity in P2P live systems has been examined in the recent literature [6–9]. In [6], the streaming capacity problem is formulated into an optimization problem, which maximizes the streaming rate that can be supported by multitree-based overlay. Sengupta et al. provide a taxonomy of sixteen problem formulations on streaming capacity, depending on whether there is a single P2P session or there are multiple concurrent sessions, whether the given topology is a full mesh graph or an arbitrary graph, whether the number of peers a node can have is bounded or not, and whether there are nonreceiver relay nodes or not [7]. Liu et al. analyze the performance bounds for minimum server load, maximum streaming rate, and minimum tree depth in tree-based P2P live systems, respectively [8]. The streaming capacity under node degree bound is investigated in [9].

Streaming capacity for a single channel in P2P VoD systems has been studied in [10–12]. In [10, 12], the streaming capacity for a single channel is formulated into an optimization problem which maximizes the streaming rate under the peer bandwidth constraints. The throughput maximization problem in a scalable P2P VoD system is studied in [13]. Helpers have been proposed in P2P systems to improve the system performance [11, 14, 15]. In P2P VoD systems, each additional helper increases the system upload capacity, thus offloading the server burden [15]. In [11], the algorithms on helper assignment and rate allocation are proposed to improve the streaming capacity for P2P VoD systems.

Cross-channel resource sharing has been recently studied in multi-channel P2P streaming systems [16–20]. In [16], Wu et al. propose an online server capacity provisioning algorithm to adjust the server capacities available to each of the concurrent channels, taking into account the number of peers, the streaming quality, and the priorities of channels. In [17], a View-Upload Decoupling (VUD) scheme is proposed to decouple what a peer uploads from what it views, bringing stability to multi-channel P2P streaming systems and enabling cross-channel resource sharing. In [18], Wu et al. develop infinite-server queueing network models to analytically study the performance of multi-channel P2P live streaming systems. In [19], the bandwidth satisfaction ratio is used to compare three bandwidth allocation schemes, namely, Naive Bandwidth allocation Approach (NBA), Passive Channel-aware bandwidth allocation Approach (PCA), and Active Channel-aware bandwidth allocation Approach (ACA), in multi-channel P2P streaming systems. In [20], Zhao et al. investigate the streaming capacity in multi-channel P2P live streaming systems when each peer can only connect to a small number of neighbors.

## 3. Streaming Capacity for an Independent-Channel P2P VoD System

In an independent-channel P2P VoD system, the peers watching the same channel form an overlay. We assume that the server upload bandwidth allocated to channel $m$, denoted by $s_m$, is predetermined. An independent-channel P2P VoD system is illustrated in Figure 1. The peers within the same overlay redistribute the video content to each other. A peer belonging to overlay $m$ only caches the video content of channel $m$, and it does not serve any video content to any peer outside overlay $m$. In other words, each overlay in an independent-channel P2P VoD System is an isolated subsystem. Therefore, the streaming capacity of a channel is independent of that of another channel. The streaming capacity problem in an independent-channel P2P VoD system is to find the streaming capacity of each individual channel, respectively. We will next focus on the streaming capacity for a single channel.

*3.1. Complete Overlay.* The streaming capacity for a single channel depends on the constructed overlay. In a P2P VoD system, each peer maintains a buffer to cache the recently received packets in order to smoothen the playback and serve

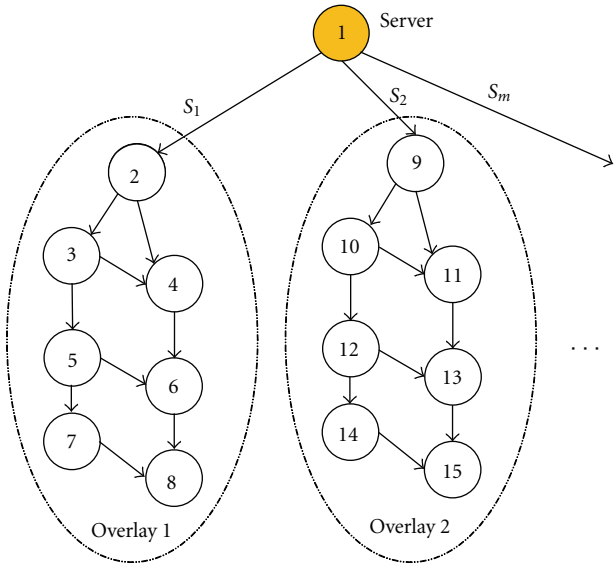



Figure 1: Illustration of an independent-channel P2P VoD system.

other peers. There is a *parent-child relationship* between two peers (e.g., peer $k$ and peer $j$) if the two peers satisfy the following two conditions: (1) peer $k$ (the parent) has an earlier playback progress than peer $j$ (the child), and (2) peer $k$ is buffering the segment(s) which are being requested by peer $j$. The parent-child relationship is illustrated in Figure 2(a). At the current moment, peer $k$ is playing segment 11, and peer $j$ is playing segment 7. Each peer can hold 5 segments in its buffer. Segment 8, held by peer $k$, is being requested by peer $j$. Therefore, peers $k$ and $j$ can establish a parent-child relationship, in which peer $k$ is the parent, and peer $j$ is the child. The server is regarded as a special peer because it contains all the segments of the video. The server has a parent-child relationship with any other peer.

A parent-child relationship is *implemented*, if an overlay link has been established from the parent to the child. A *complete overlay* for a channel is defined as the overlay in which all of the available parent-child relationships have been implemented, and an *incomplete overlay* for a channel is defined as the overlay in which only a part of the available parent-child relationships have been implemented [12]. An example of complete overlay is illustrated in Figure 2(b). Compared to the incomplete overlay, the complete overlay contains more overlay links, thus supporting a higher streaming capacity. Therefore, we will formulate the streaming capacity problem based on the complete overlay. The method for overlay construction is not a study focus of this paper. A complete overlay can be constructed using the existing approaches [2].

*3.2. Streaming Capacity Problem for a Single Channel.* The set of the overlays (channels) in the independent-channel P2P VoD system is denoted by $\mathbf{M}$. Since each channel is independent in terms of resources, we can just study the streaming capacity of channel $m$ ($\forall m \in \mathbf{M}$).

Overlay $m$ in a P2P VoD system can be modeled as a directed graph $\mathbf{G}^{(\mathbf{m})} = (\mathbf{N}^{(\mathbf{m})}, \mathbf{L}^{(\mathbf{m})})$, where $\mathbf{N}^{(\mathbf{m})}$ is the set of nodes and $\mathbf{L}^{(\mathbf{m})}$ is the set of directed overlay links. Peer 1 is defined as the server. The relationship between a node and its outgoing links is represented with a matrix $\mathbf{A}^{(\mathbf{m})+}$, whose elements are given by

$$a_{il}^{(m)+} = \begin{cases} 1, & \text{if link } l \text{ is an outgoing link from node } i, \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

The relationship between a node and its incoming links is represented with a matrix $\mathbf{A}^{(\mathbf{m})-}$, whose elements are given by

$$a_{il}^{(m)-} = \begin{cases} 1, & \text{if link } l \text{ is an incoming link into node } i, \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

The upload capacity of peer $i$ is denoted by $O_i^{(m)}$. The server is denoted as Peer 1. The server upload bandwidth $s_m$ for channel $m$ is actually the upload capacity of Peer 1, which is given by $O_1^{(m)} = s_m$. Upload capacity is typically the bottleneck in P2P systems. We consider *upload constraint* at peer $i$ ($\forall i \in \mathbf{N}^{(\mathbf{m})}$), which is given by $\sum_{l \in \mathbf{L}^{(\mathbf{m})}} a_{il}^{(m)+} x_l^{(m)} \le O_i^{(m)}$ where $x_l^{(m)}$ is the link rate at link $l$ of overlay $m$. The upload constraint represents that the total outgoing rate from peer $i$ is no larger than its upload capacity $O_i^{(m)}$. The streaming rate for channel $m$ is denoted by $r^{(m)}$. Each peer except the server (e.g., Peer 1) receives the streaming rate $r^{(m)}$, which can be expressed by $\sum_{l \in \mathbf{L}} a_{il}^{(m)-} x_l^{(m)} = f_i^{(m)} r^{(m)}$, for all $i \in \mathbf{N}^{(\mathbf{m})}$, where $f_i^{(m)} = 0$ if $i = 1$, or $f_i^{(m)} = 1$ otherwise. The *download constraint* is given by $r^{(m)} \le \min_{i \in \mathbf{N}^{(\mathbf{m})}} I_i^{(m)}$ where $I_i^{(m)}$ is the download capacity of peer $i$. From the download constraint, we can see that the maximal streaming rate is limited by the minimal download capacity among the peers. Therefore, the users with a very low download bandwidth should not be admitted into the P2P VoD system in order for maintaining a high streaming capacity.

The streaming capacity for channel $m$ is stated as to maximize the streaming rate $r^{(m)}$ that can be received by every user in channel $m$ by optimizing the streaming rate $r^{(m)}$ and the link rate $x_l^{(m)}$ ($\forall l \in \mathbf{L}^{(\mathbf{m})}$) under the upload constraint at each peer. Mathematically, the problem can be formulated as follows:

$$\begin{aligned} \text{maximize} \quad & r^{(m)} \\ \text{subject to} \quad & \sum_{l \in \mathbf{L}^{(\mathbf{m})}} a_{il}^{(m)-} x_l^{(m)} = f_i^{(m)} r^{(m)}, \quad \forall i \in \mathbf{N}^{(\mathbf{m})}, \\ & \sum_{l \in \mathbf{L}^{(\mathbf{m})}} a_{il}^{(m)+} x_l^{(m)} \le O_i^{(m)}, \quad \forall i \in \mathbf{N}^{(\mathbf{m})}, \\ & x_l^{(m)} \ge 0, \quad \forall l \in \mathbf{L}^{(\mathbf{m})}, \\ & 0 \le r^{(m)} \le \min_{i \in \mathbf{N}^{(\mathbf{m})}} I_i^{(m)}. \end{aligned} \tag{3}$$

The optimization problem in (3) is a Linear Programming (**LP**). It can be solved in a centralized way using

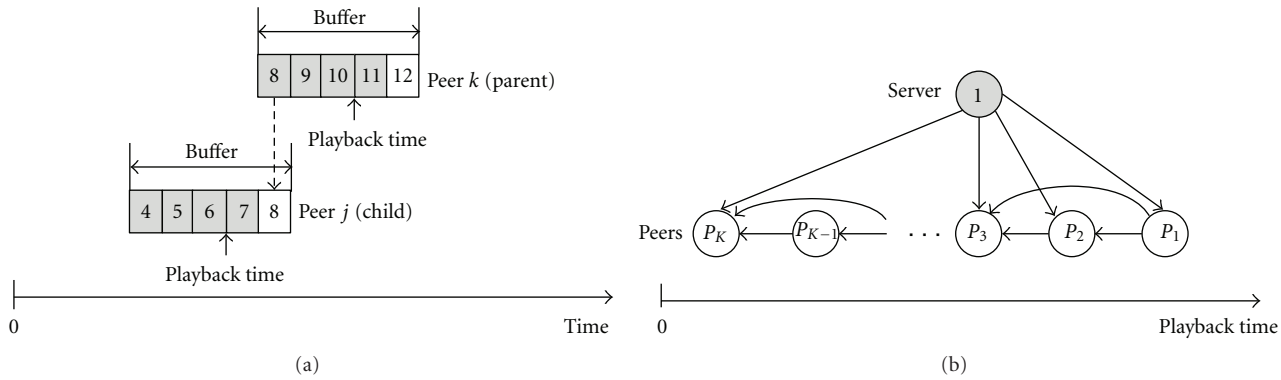



FIGURE 2: Overlay construction for a single channel in an independent-channel P2P VoD system: (a) the parent-child relationship between peer $k$ and peer $j$, and (b) the complete overlay.

the interior point method [21]. However, the centralized solution is not scalable. Therefore, we will develop a distributed algorithm using the primal-dual method to solve the streaming capacity problem.

*3.3. Distributed Algorithm.* We will use primal-dual method [22] to develop a distributed algorithm for the optimization problem (3). However, the objective function in problem (3) is not strictly convex with respect to the optimization variables $r^{(m)}$ and $x_l^{(m)}$ ($\forall l \in \mathbf{L^{(m)}}$). Therefore, the corresponding dual function is nondifferentiable, and the optimal values of $r^{(m)}$ and $x_l^{(m)}$ ($\forall l \in \mathbf{L^{(m)}}$) are not immediately available. We first convert the the original optimization problem (3) to an equivalent minimization problem by changing the objective from maximizing $r^{(m)}$ to minimizing $-r^{(m)}$. We then add a quadratic regularization term for the streaming rate $r^{(m)}$ and each link rate $x_l^{(m)}$ ($\forall l \in \mathbf{L^{(m)}}$) to make the objective function strictly convex. Finally, the optimization problem (3) is approximated to the following:

$$\begin{aligned}
\text{minimize} \quad & -r^{(m)} + \varepsilon\left(r^{(m)}\right)^2 + \varepsilon \sum_{l\in\mathbf{L^{(m)}}} \left(x_l^{(m)}\right)^2 \\
\text{subject to} \quad & \sum_{l\in\mathbf{L^{(m)}}} a_{il}^{(m)-} x_l^{(m)} = f_i^{(m)} r^{(m)}, \quad \forall i \in \mathbf{N^{(m)}}, \\
& \sum_{l\in\mathbf{L^{(m)}}} a_{il}^{(m)+} x_l^{(m)} \le O_i^{(m)}, \quad \forall i \in \mathbf{N^{(m)}}, \\
& x_l^{(m)} \ge 0, \quad \forall l \in \mathbf{L^{(m)}}, \\
& 0 \le r^{(m)} \le \min_{i\in\mathbf{N^{(m)}}} I_i^{(m)},
\end{aligned} \tag{4}$$

where $(\varepsilon(r^{(m)})^2 + \varepsilon\sum_{l\in\mathbf{L^{(m)}}}(x_l^{(m)})^2)$ is the sum of the regularization terms, and $\varepsilon(\varepsilon > 0)$ is called the *regularization factor*. When $\varepsilon$ is small enough, the solution for the problem in (4) is arbitrarily close to the solution for the original streaming capacity problem (3).

Let us denote by $(r^{(m)*}, x_l^{(m)*}\ (\forall l \in \mathbf{L^{(m)}}))$ the optimal solution to the original optimization problem (3), and $(r^{(m)\$}, x_l^{(m)\$}\ (\forall l \in \mathbf{L^{(m)}}))$ the optimal solution to the approximated optimization problem (4). Based on the original optimization problem (3), we have $r^{(m)*} \ge r^{(m)\$}$. Based on the approximated optimization problem (4), we have $-r^{(m)\$} + \varepsilon(r^{(m)\$})^2 + \varepsilon\sum_{l\in\mathbf{L^{(m)}}}(x_l^{(m)\$})^2 \le -r^{(m)*} + \varepsilon(r^{(m)*})^2 + \varepsilon\sum_{l\in\mathbf{L^{(m)}}}(x_l^{(m)*})^2$. Therefore, we can get $0 \le r^{(m)*} - r^{(m)\$} \le \varepsilon((r^{(m)*})^2 - (r^{(m)\$})^2 + \sum_{l\in\mathbf{L^{(m)}}}(x_l^{(m)*})^2 - \sum_{l\in\mathbf{L^{(m)}}}(x_l^{(m)\$})^2)$, from which we show that the streaming capacity obtained from the approximated optimization problem (4) with a small regularization factor can approach closely to the truly maximal streaming rate obtained from the original optimization problem (3).

The optimization problem (4) is a convex optimization problem with a strictly convex objective function and the linear constraints [23]. We introduce dual variables ($u_i, v_i$, $\forall i \in \mathbf{N^{(m)}}$) to formulate the Lagrangian corresponding to the primal problem (4) as below:

$$\begin{aligned}
&L\left(\mathbf{x}^{(m)}, r^{(m)}, \mathbf{u}, \mathbf{v}\right) \\
&\quad = -r^{(m)} + \varepsilon\left(r^{(m)}\right)^2 + \varepsilon \sum_{l\in\mathbf{L^{(m)}}} \left(x_l^{(m)}\right)^2 \\
&\qquad + \sum_{i\in\mathbf{N^{(m)}}} u_i \left( \sum_{l\in\mathbf{L^{(m)}}} a_{il}^{(m)-} x_l^{(m)} - f_i^{(m)} r^{(m)} \right) \\
&\qquad + \sum_{i\in\mathbf{N^{(m)}}} v_i \left( \sum_{l\in\mathbf{L^{(m)}}} a_{il}^{(m)+} x_l^{(m)} - O_i^{(m)} \right) \\
&\quad = \varepsilon\left(r^{(m)}\right)^2 - r^{(m)} - r^{(m)} \sum_{i\in\mathbf{N^{(m)}}} u_i f_i^{(m)} \\
&\qquad + \sum_{l\in\mathbf{L^{(m)}}} \left( \varepsilon\left(x_l^{(m)}\right)^2 \right. \\
&\qquad\qquad \left. + x_l^{(m)} \sum_{i\in\mathbf{N^{(m)}}} \left( u_i a_{il}^{(m)-} + v_i a_{il}^{(m)+} \right) \right) \\
&\qquad - \sum_{i\in\mathbf{N^{(m)}}} v_i O_i^{(m)}.
\end{aligned} \tag{5}$$

Let $I_{\min}^{(m)} = \min_{i\in\mathbf{N^{(m)}}} I_i^{(m)}$. The Lagrange dual function $G(\mathbf{u}, \mathbf{v})$ is the minimum value of the Lagrangian over the vector of link rates $\mathbf{x}^{(m)}$ and the streaming rate $r^{(m)}$:

$$G(\mathbf{u}, \mathbf{v}) = \min_{\left(\mathbf{x}^{(m)}\ge\mathbf{0}, 0\le r^{(m)}\le I_{\min}^{(m)}\right)} \left\{ L\left(\mathbf{x}^{(m)}, r^{(m)}, \mathbf{u}, \mathbf{v}\right) \right\}. \tag{6}$$

The Lagrange dual problem is to maximize the Lagrange dual function. That is,

$$\begin{aligned} &\text{maximize} && G(\mathbf{u},\mathbf{v}) \\ &\text{subject to} && v_i \geq 0, \qquad \forall i \in \mathbf{N^{(m)}}. \end{aligned} \tag{7}$$

We use subgradient method [24] to solve the Lagrange dual problem (7). The dual variables $u_i^{(k+1)}$ and $v_i^{(k+1)}$ at the $(k+1)$th iteration are updated, respectively, by

$$\begin{aligned} u_i^{(k+1)} &= u_i^{(k)} - \theta^{(k)}\left(f_i^{(m)}\left(r^{(m)}\right)^{(k)} - \sum_{l\in\mathbf{L^{(m)}}} a_{il}^{(m)-}\left(x_l^{(m)}\right)^{(k)}\right), \\ v_i^{(k+1)} &= \max\left\{0, v_i^{(k)} - \theta^{(k)}\left(O^{(m)} - \sum_{l\in\mathbf{L^{(m)}}} a_{il}^{(m)+}\left(x_l^{(m)}\right)^{(k)}\right)\right\}, \end{aligned} \tag{8}$$

where $\theta^{(k)} > 0$ is the step size at the $k$th iteration. The algorithm is guaranteed to converge to the optimal value for a step-size sequence satisfying the nonsummable diminishing rule [24]:

$$\lim_{k\to\infty}\theta^{(k)} = 0, \qquad \sum_{k=1}^{\infty}\theta^{(k)} = \infty. \tag{9}$$

The update of the streaming rate $(r^{(m)})^{(k)}$ at the $k$th iteration is given by

$$\left(r^{(m)}\right)^{(k)} = \min\left\{\max\left\{0, \frac{1+\sum_{i\in\mathbf{N^{(m)}}} u_i^{(k)} f_i^{(m)}}{2\varepsilon}\right\}, I_{\min}^{(m)}\right\}. \tag{10}$$

At the $k$th iteration, the link rate $(x_l^{(m)})^{(k)}$ at link $l$ can be calculated from the dual variables:

$$\left(x_l^{(m)}\right)^{(k)} = \max\left\{0, \frac{-\sum_{i\in\mathbf{N^{(m)}}}\left(u_i^{(k)} a_{il}^{(m)-} + v_i^{(k)} a_{il}^{(m)+}\right)}{2\varepsilon}\right\}. \tag{11}$$

As shown in (10) and (11), the optimization variables are decomposed. The optimization of the streaming rate is executed at the server as follows. At the $k$th iteration, the server collects the dual variable $u_i^{(k)}$ from peer $i$ ($\forall i \in \mathbf{N^{(m)}}$), respectively, and then computes the streaming rate $(r^{(m)})^{(k)}$. The optimization of each link rate is executed at each peer as follows. At the $k$th iteration, each peer only collects the dual variable $u_j^{(k)}$ from each of its children, and then computes the link rate for each of the outgoing links.

A P2P system is inherently dynamic. Peers may join or leave the system at any time. Therefore, the streaming capacity for a single channel varies with time. To handle the time-varying streaming capacity, the server can encode the video using a scalable coding scheme and then adjust the output video rate adaptive to the current streaming capacity.

## 4. Streaming Capacity for a Correlated-Channel P2P VoD System

In a correlated-channel P2P VoD system, the resources among different channels can be shared with each other. The average streaming capacity in a correlated-channel P2P VoD system can be obtained by optimizing both the intra-channel resource allocation and the cross-channel resource allocation. The optimization of intra-channel resource allocation for a single channel has been presented in Section 3. In this section, we will focus on the cross-channel resource allocation. We will first optimize the server upload allocation among channels to maximize the average streaming capacity in Section 4.1, and then further utilize cross-channel peer upload bandwidth to improve the average streaming capacity in Section 4.2.

*4.1. Optimization of Server Upload Allocation among Channels.* The server upload allocated for channel $m$ is denoted by $s_m$. In an independent-channel P2P VoD system, the server upload allocation for each channel is predetermined. Therefore, the server upload bandwidth is not utilized in an optimal way. In this subsection, we treat $\{s_m, \forall m \in \mathbf{M}\}$ as variables and optimize them to maximize the average streaming capacity for a correlated-channel P2P VoD system.

The optimization of server upload allocation for a correlated-channel P2P VoD system is stated as to maximize the average streaming rate by optimizing the server upload allocation, the link rates, and the streaming rate, for each channel, subject to the upload constraint at each peer. Since maximizing the average streaming rate is equivalent to minimizing the negative average streaming rate, we formulate the problem into a minimization problem as follows:

$$\begin{aligned} &\text{minimize} && -\sum_{m\in\mathbf{M}} p_m r^{(m)} \\ &\text{subject to} && \sum_{l\in\mathbf{L^{(m)}}} a_{il}^{(m)-} x_l^{(m)} = f_i^{(m)} r^{(m)}, \\ & && \qquad \forall i \in \mathbf{N^{(m)}}, \forall m \in \mathbf{M}, \\ & && \sum_{l\in\mathbf{L^{(m)}}} a_{il}^{(m)+} x_l^{(m)} \leq O_i^{(m)}, \\ & && \qquad \forall i \in \mathbf{N^{(m)}}, \forall m \in \mathbf{M}, \\ & && x_l^{(m)} \geq 0, \qquad \forall l \in \mathbf{L^{(m)}}, \\ & && \sum_{m\in\mathbf{M}} s_m \leq s_T, \\ & && s_m \geq 0, \qquad \forall m \in \mathbf{M}, \\ & && 0 \leq r^{(m)} \leq \min_{i\in\mathbf{N^{(m)}}} I_i^{(m)}, \qquad \forall m \in \mathbf{M}, \end{aligned} \tag{12}$$

where $p_m$ is the priority for channel $m$, and $s_T$ is the server upload capacity. The optimization problem in (12) is an LP. The optimization variables in the optimization problem (12) are the server upload allocation $s_m$, the link rate vector $\mathbf{x^{(m)}}$, and the streaming rate $r^{(m)}$, for channel $m$, for all $m \in \mathbf{M}$. The number of the optimization variables is increased with the number of the channels and the number of the links in

each overlay. In order to solve the optimization efficiently, we use dual decomposition [22] to decompose the optimization problem (12) into multiple subproblems, each of which is associated with a channel.

We introduce a dual variable $\lambda$ for the inequality constraint $\sum_{m\in\mathbf{M}} s_m \le s_T$. The Lagrangian corresponding to the primal problem (12) is given by $L(\mathbf{s},\mathbf{x},\mathbf{r},\lambda) = -\Sigma_{m\in\mathbf{M}} p_m r^{(m)} + \lambda(\sum_{m\in\mathbf{M}} s_m - s_T)$. Then, the Lagrange dual function [23] is given by $G(\lambda) = \min L(\mathbf{s},\mathbf{x},\mathbf{r},\lambda) = \sum_{m\in\mathbf{M}} \min(-p_m r^{(m)} + \lambda s_m) - \lambda s_T$.

The Lagrange dual problem is to maximize the Lagrange dual function [23]. That is,

$$\begin{aligned} &\text{maximize} && G(\lambda) \\ &\text{subject to} && \lambda \ge 0. \end{aligned} \tag{13}$$

Subgradient method [24] is used to solve the Lagrange dual problem (13). The dual variable $\lambda$ is updated at the $(k+1)$th iteration by $\lambda^{(k+1)} = \max\{0, \lambda^{(k)} - \theta^{(k)}(s_T - \sum_{m\in\mathbf{M}} s_m^{(k)})$ where $\theta^{(k)}$ is the step size at the $k$th iteration. In order to guarantee the convergence, the sequence of the step sizes is required to satisfy the nonsummable diminishing rule in (9).

At the $k$th iteration, the primal variables $(s_m, \mathbf{x}^{(\mathbf{m})}, r^{(m)})$ for channel $m$ are obtained by solving the following optimization problem:

$$\begin{aligned} &\text{minimize} && -p_m r^{(m)} + \lambda s_m \\ &\text{subject to} && \sum_{l\in\mathbf{L}^{(\mathbf{m})}} a_{il}^{(m)-} x_l^{(m)} = f_i^{(m)} r^{(m)}, \quad \forall i \in \mathbf{N}^{(\mathbf{m})}, \\ & && \sum_{l\in\mathbf{L}^{(\mathbf{m})}} a_{il}^{(m)+} x_l^{(m)} \le O_i^{(m)}, \quad \forall i \in \mathbf{N}^{(\mathbf{m})}, \\ & && x_l^{(m)} \ge 0, \quad \forall l \in \mathbf{L}^{(\mathbf{m})}, \\ & && 0 \le r^{(m)} \le \min_{i\in\mathbf{N}^{(\mathbf{m})}} I_i^{(m)}, \quad s_m \ge 0. \end{aligned} \tag{14}$$

The optimization problem (12) is decomposed into $|M|$ subproblems where $|M|$ is the number of the channels. Subproblem $m$, represented in (14), is associated with channel $m$. Subproblem $m$, for all $m \in \mathbf{M}$, can be solved with a distributed algorithm.

*4.2. Cross-Channel Sharing of Peer Upload Bandwidth.* Though the server upload allocation among channels is optimized, the cross-channel resources have not yet been fully utilized. In each channel, there are a number of underutilized peers, which can be utilized by the other channels. For example, the leaf nodes of overlay $m$ contribute zero upload bandwidth to the channel because they have no outgoing links. We can utilize the upload bandwidth of the leaf nodes in overlay $m$ to serve the peers in another channel (e.g., channel $k$), thus improving the streaming capacity of channel $k$. However, it is challenging to establish the cross-channel links to enable the cross-channel sharing of peer upload bandwidth.

In this paper, we propose a scheme for cross-channel peer upload sharing. We introduce the concept of *cross-channel helpers*. A *cross-channel helper* is the peer who uses its remaining upload bandwidth to help other peers in another channel.

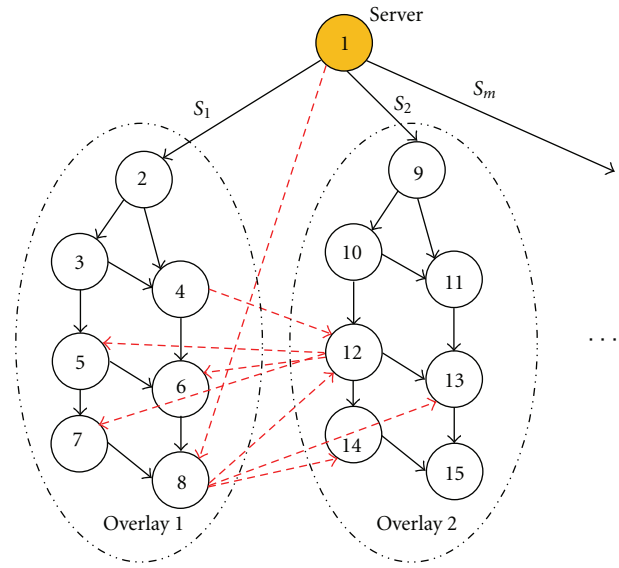



FIGURE 3: Illustration of a correlated-channel P2P VoD system.

Cross-channel helpers are chosen from the peers who have a remaining upload bandwidth greater than a threshold.

Suppose that peer $i$ in channel $m$ is a cross-channel helper serving segment $j$ of channel $k$, and it has a remaining upload bandwidth $b_i$. We denote the set of peers watching segment $j$ in channel $k$ by $\mathbf{H}_j^{(k)}$. Peer $i$ can download segment $j$ of channel $k$ from the server or the peers who are buffering segment $j$, at a rate $R_i^{\mathrm{IN}}$, and then output it to peer $h$, for all $h \in \mathbf{H}_j^{(k)}$, at a rate $R_i^h$, respectively. The *bandwidth gain* for peer $i$ is defined as $g_i = (\sum_{h\in\mathbf{H}_j^{(k)}} R_i^h)/R_i^{\mathrm{IN}}$. In order to maximize the bandwidth gain $g_i$ under the flow constraint $R_i^h \le R_i^{\mathrm{IN}}$, for all $h \in \mathbf{H}_j^{(k)}$, and the bandwidth constraint $\sum_{h\in\mathbf{H}_j^{(k)}} R_i^h \le b_i$, the optimal download rate at peer $i$ is $R_i^{\mathrm{IN}*} = b_i/|\mathbf{H}_j^{(k)}|$ where $|\mathbf{H}_j^{(k)}|$ is the number of the peers watching segment $j$ of channel $k$, and the optimal outgoing rate to peer $h$, for all $h \in \mathbf{H}_j^{(k)}$, is $R_i^{h*} = R_i^{\mathrm{IN}*}$. The bandwidth gain should be larger than 1, otherwise, the cross-channel helper consumes a larger bandwidth than it contributes. Figure 3 illustrates a correlated-channel P2P VoD system. As shown in Figure 3, peer 8 in overlay 1 is a cross-channel helper, who downloads a segment from the server and then forwards it to peers 12–14 in overlay 2; peer 12 in overlay 2 is also a cross-channel helper, who downloads a segment from peer 4 in overlay 1 and then forwards it to peers 5–7 in overlay 1.

The proposed scheme for cross-channel peer upload sharing is described as follows.

(1) Channel $m$ ($\forall m \in \mathbf{M}$) finds a partner, channel $k$ ($k \ne m$, $k \in \mathbf{M}$), to help each other, by using a *resource-balancing scheme*, which is described as follows. (i) Calculate $B_m^{\mathrm{re}}$, the amount of the total remaining bandwidth, for channel $m$ ($\forall m \in \mathbf{M}$), the average amount of the total remaining bandwidth is given by $B_{\mathrm{avg}}^{\mathrm{re}} = (\sum_{m\in\mathbf{M}} B_m^{\mathrm{re}})/|\mathbf{M}|$ where $|\mathbf{M}|$

represents the number of channels. (ii) Given the set of unchosen channels $\mathbf{M}^{\text{un}}$, the partner of channel $m$ is determined by $k = \arg_{j \in \mathbf{M}^{\text{un}}} \min((1/2)(B_j^{\text{re}} + B_m^{\text{re}}) - B_{\text{avg}}^{\text{re}})^2$.

(2) Determine the set of cross-channel helpers in channel $m$, denoted by $\mathbf{Y}^{(m)}$, by choosing the peers who have a remaining upload bandwidth greater than a threshold $b$th. Sort $\mathbf{Y}^{(m)}$ in a descending order based on the remaining upload bandwidth.

(3) Determine the *demanding segment set* in channel $k$, denoted by $\mathbf{W}^{(k)}$, by choosing $|\mathbf{Y}^{(m)}|$ segments which are watched by the largest number of peers. Sort $\mathbf{W}^{(k)}$ in a descending order based on the number of watching peers.

(4) Assign the $i$th cross-channel helper in $\mathbf{Y}^{(m)}$ to serve the peers watching the $i$th segment in $\mathbf{W}^{(k)}$. Determine the incoming rate and outgoing rates at the $i$th cross-channel helper to maximize the bandwidth gain.

(5) After allocating the rate for each of the cross-channel links as in Step (4), revise the optimization problem (12) by integrating the cross-channel link rates and then solve it to obtain the optimal server upload allocation for each channel and the optimal link rates within each overlay.

Peer dynamics have an impact on the streaming capacity in correlated-channel P2P VoD systems. First, the peers may leave or join a channel dynamically. Second, the cross-channel helpers may leave the channel, which causes the disconnection of the cross-channel links. To handle the dynamic conditions, the optimizations of intra-channel resource allocation and cross-channel resource allocation need to be performed in a discrete-time manner, in which the peers and the overlay are assumed to remain unchanged during a time slot, and the algorithms for resource allocation are performed at the beginning of each time slot.

## 5. Simulations

In the simulations, we use two classes of peers: cable/DSL peers and Ethernet peers. Cable/DSL peers take 85% of the total peer population with download capacity uniformly distributed between 0.9 Mbps and 1.5 Mbps and upload capacity uniformly distributed between 0.3 Mbps and 0.6 Mbps. Ethernet peers take the remaining 15% of the total peer population with both upload and download capacities uniformly distributed between 1.5 Mbps and 3.0 Mbps. The length of the video is 60 minutes, which is evenly divided into 60 segments. Each peer maintains a buffer with a capacity of 5 segments. The playback time of each peer is randomly distributed between 0 and 60 minutes. The priorities for all channels are equal.

Figure 4 compares the average streaming capacity between the *equal server upload* scheme and the *optimized server upload* scheme. In the *equal server upload* scheme, each channel is allocated an equal server upload, the link rates within each channel are optimized by solving the optimization problem (3). In the *optimized server upload* scheme, the server upload for each channel and the link rates within each channel are jointly optimized by solving the optimization problem (12). The *equal server upload* scheme considers only intra-channel resource allocation [10], while the *optimized server upload* scheme considers both intra-channel and cross-channel resource allocation. The number of the peers in a channel is uniformly distributed between 10 and 150. The server upload capacity is 45.0 Mbps. The optimized server upload scheme improves the average streaming capacity by 14.1% in average compared to the equal server upload scheme.

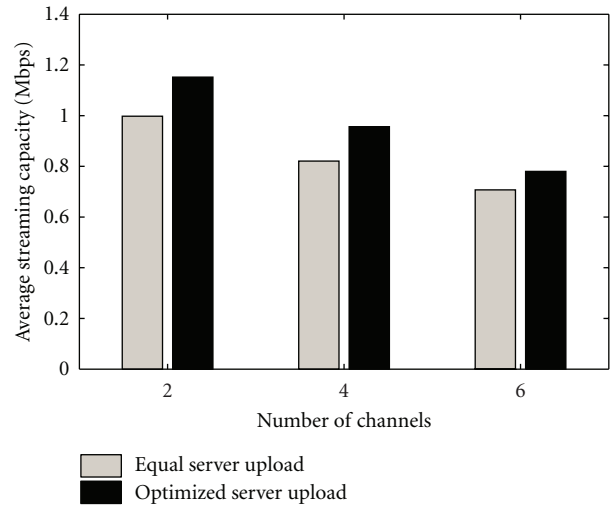



Figure 4: Comparison of average streaming capacity with different number of channels.

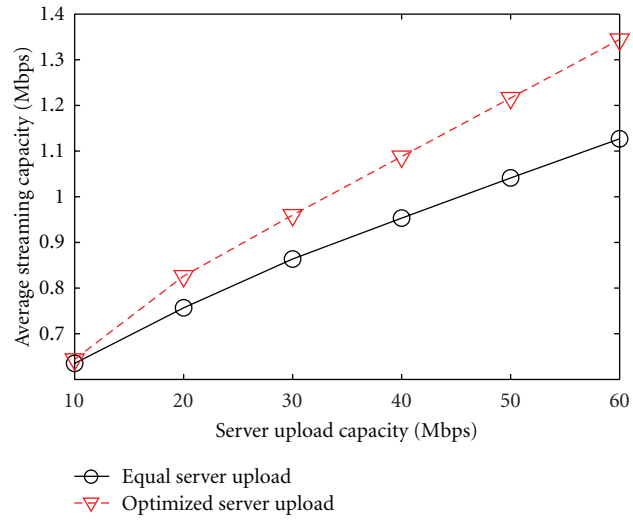



Figure 5: Impact of server upload capacity to the average streaming capacity.

We show in Figure 5 the impact of server upload capacity to the average streaming capacity for a P2P VoD system with 2 channels. The first channel has 40 peers, and the

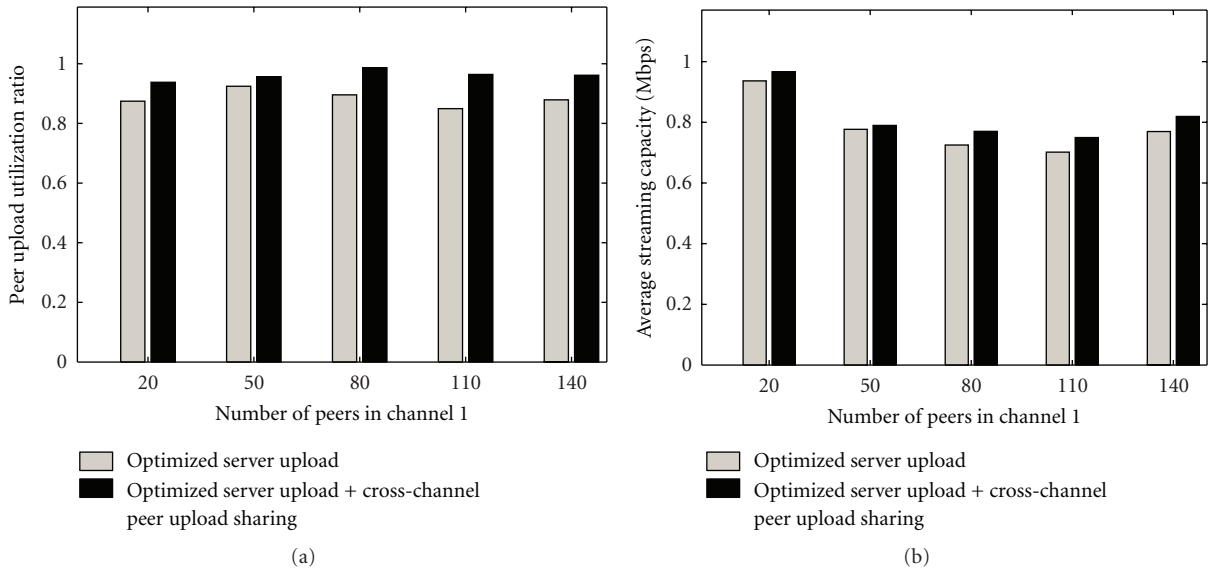



Figure 6: Performance improvement brought by cross-channel peer upload sharing: (a) peer upload utilization ratio, and (b) average streaming capacity.

second one has 120 peers. As shown in Figure 5, the average streaming capacity is increased with the server upload capacity. The improvement brought by the optimized server upload scheme is larger when the server upload capacity is larger.

Joint consideration of server upload optimization and cross-channel peer upload sharing can improve the performance in a correlated P2P VoD system, as shown in Figure 6. There are two channels with total 200 peers in the P2P VoD system. The server upload capacity is 20.0 Mbps. We vary the number of the peers in channel 1 from 20 to 140 and evaluate two metrics: peer upload utilization ratio and average streaming capacity. *Peer upload utilization ratio* is defined by $\beta = \Sigma_{i\in\mathbf{N},i\neq 1} v_i / \Sigma_{i\in\mathbf{N},i\neq 1} O_i$ where $\Sigma_{i\in\mathbf{N},i\neq 1} v_i$ represents the sum of the outgoing rates from all peers except the server, and $\Sigma_{i\in\mathbf{N},i\neq 1} O_i$ represents the sum of the upload capacities of all peers except the server. Due to a better utilization of cross-channel resources, joint consideration of server upload optimization and cross-channel peer upload sharing improves the peer upload utilization ratio by 7.7% in average, as shown in Figure 6(a), and improves the average streaming capacity by 0.04 Mbps in average, as shown in Figure 6(b), compared to *optimized server upload scheme*.

## 6. Conclusion

In this paper, we studied the streaming capacity problem for multi-channel P2P VoD systems. Depending on the resource correlation, multi-channel P2P VoD systems can be categorized into independent-channel P2P VoD systems and correlated-channel P2P VoD systems. Since there is no resource correlation among channels in an independent-channel P2P VoD system, we just need to find the streaming capacity for each individual channel, respectively. We formulated the streaming capacity problem for a single channel into an LP problem and solved it with a distributed algorithm. In a correlated-channel P2P VoD system, we optimized both the intra-channel resource allocation and cross-channel resource allocation to improve the streaming capacity. In order to better utilize the cross-channel resource, we first optimized the server upload allocation among channels to maximize the average streaming capacity and then introduced cross-channel helpers to enable cross-channel sharing of peer upload bandwidth. We demonstrated in the simulations that the correlated-channel P2P VoD systems with both intra-channel and cross-channel resource allocation can achieve a higher average streaming capacity compared to the independent-channel P2P VoD systems with only intra-channel resource allocation.

## References

[1] J. Li, "PeerStreaming: an on-demand peer-to-peer media streaming solution based on a receiver-driven streaming protocol," in *Proceedings of the IEEE 7th Workshop on Multimedia Signal Processing (MMSP '05)*, pp. 1–4, November 2005.

[2] H. Chi, Q. Zhang, J. Jia, and X. Shen, "Efficient search and scheduling in P2P-based media-on-demand streaming service," *IEEE Journal on Selected Areas in Communications*, vol. 25, no. 1, pp. 119–130, 2007.

[3] S. Annapureddy, C. Gkantsidis, and P. Rodriguez, "Providing video-on-demand using peer-to-peer networks," in *Proceedings of the International Conference on World Wide Web*, 2006.

[4] C. Huang, J. Li, and K. W. Ross, "Peer-assisted VoD: making Internet video distribution cheap," in *Proceedings of the of IPTPS*, 2007.

[5] Y. Huang, T. Z. J. Fu, D. M. Chiu, J. C. S. Lui, and C. Huang, "Challenges, design and analysis of a large-scale p2p-vod system," in *Proceedings of the ACM SIGCOMM Conference on Data Communication (SIGCOMM '08)*, pp. 375–388, August 2008.

[6] S. Sengupta, S. Liu, M. Chen, M. Chiang, J. Li, and P. A. Chou, "Streaming capacity in peer-to-peer networks with topology constraints," Microsoft Research Technical Report, 2008.

[7] S. Sengupta, S. Liu, M. Chen, M. Chiang, J. Li, and P. A. Chou, "Peer-to-peer streaming capacity," *IEEE Transactions on Information Theory*, vol. 57, no. 8, pp. 5072–5087, 2011.

[8] S. Liu, R. Zhang-Shen, W. Jiang, J. Rexford, and M. Chiang, "Performance bounds for Peer-assisted live streaming," in *Proceedings of the ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS '08)*, pp. 313–324, June 2008.

[9] S. Liu, M. Chen, S. Sengupta, M. Chiang, J. Li, and P. A. Chou, "P2P streaming capacity under node degree bound," in *Proceedings of the 30th IEEE International Conference on Distributed Computing Systems (ICDCS '10)*, pp. 587–598, June 2010.

[10] Y. He and L. Guan, "Streaming capacity in P2P VoD systems," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS '09)*, pp. 742–745, May 2009.

[11] Y. He and L. Guan, "Improving the streaming capacity in P2P VoD systems with helpers," in *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME '09)*, pp. 790–793, July 2009.

[12] Y. He and L. Guan, "Solving streaming capacity problems in P2P VoD systems," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 20, no. 11, pp. 1638–1642, 2010.

[13] Y. He, I. Lee, and L. Guan, "Distributed throughput maximization in P2P VoD applications," *IEEE Transactions on Multimedia*, vol. 11, no. 3, pp. 509–522, 2009.

[14] J. Wang, C. Yeo, V. Prabhakaran, and K. Ramchandran, "On the role of helpers in peer-to-peer file download systems: design, analysis and simulation," in *Proceedings of the IPTPS*, 2007.

[15] P. Garbacki, D. Epema, J. Pouwelse, and M. van Steen, "Offloading servers with collaborative video on demand," in *Proceedings of the IPTPS*, 2008.

[16] C. Wu, B. Li, and S. Zhao, "Multi-channel live P2P streaming: refocusing on servers," in *Proceedings of the 27th IEEE Communications Society Conference on Computer Communications (INFOCOM '08)*, pp. 2029–2037, April 2008.

[17] D. Wu, C. Liang, Y. Liu, and K. Ross, "View-upload decoupling: a redesign of multi-channel P2P video systems," in *Proceedings of the IEEE 28th Conference on Computer Communications (INFOCOM '09)*, pp. 2726–2730, April 2009.

[18] D. Wu, Y. Liu, and K. W. Ross, "Queuing network models for multi-channel P2P live streaming systems," in *Proceedings of the IEEE 28th Conference on Computer Communications (INFOCOM '09)*, pp. 73–81, April 2009.

[19] M. Wang, L. Xu, and B. Ramamurthy, "Linear programming models for multi-channel P2P streaming systems," in *Proceedings of the IEEE Conference on Computer Communications (INFOCOM '10)*, 2010.

[20] C. Zhao, X. Lin, and C. Wu, "The streaming capacity of sparsely-connected P2P systems with distributed control," in *Proceedings of the IEEE Conference on Computer Communications (INFOCOM '11)*, pp. 1449–1457, 2011.

[21] R. J. Vanderbei, *Linear Programming: Foundations and Extensions*, Springer, 2nd edition, 2001.

[22] D. Palomar and M. Chiang, "A tutorial on decomposition methods and distributed network resource allocation," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 8, pp. 1439–1451, 2006.

[23] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, 2004.

[24] D. P. Bertsekas, A. Nedic, and A. E. Ozdaglar, *Convex Analysis and Optimization*, Athena Scientific, 2003.

Hindawi Publishing Corporation
International Journal of Digital Multimedia Broadcasting
Volume 2012, Article ID 431958, 13 pages
doi:10.1155/2012/431958

*Research Article*

# A Gain-Computation Enhancements Resource Allocation for Heterogeneous Service Flows in IEEE 802.16 m Mobile Networks

**Wafa Ben Hassen and Meriem Afif**

*Mediatron: Research Unit on Radio Communication and Multimedia Networks, Higher School of Communication of Tunis (Sup'com), Carthage University, Cité Technologique des Communications, Route de Raoued Km 3.5, 2083 El Ghazala, Ariana, Tunisia*

Correspondence should be addressed to Wafa Ben Hassen, wafa.benhassen@hotmail.fr and
Meriem Afif, mariem.afif@supcom.rnu.tn

Received 30 April 2011; Revised 6 October 2011; Accepted 26 October 2011

Academic Editor: Yifeng He



This paper deals with radio resource allocation in fourth generation (4G) wireless mobile networks based on Orthogonal Frequency Division Multiple Access (OFDMA) as an access method. In IEEE 802.16 m standard, a contiguous method for subchannel construction is adopted in order to reduce OFDMA system complexity. In this context, we propose a new subchannel gain computation method depending on frequency responses dispersion. This method has a crucial role in the resource management and optimization. In a single service access, we propose a dynamic resource allocation algorithm at the physical layer aiming to maximize the cell data rate while ensuring fairness among users. In heterogeneous data traffics, we study scheduling in order to provide delay guaranties to real-time services, maximize throughput of non-real-time services while ensuring fairness to users. We compare performances to recent existing algorithms in OFDMA systems showing that proposed schemes provide lower complexity, higher total system capacity, and fairness among users.

## 1. Introduction

In fourth Generation (4G) wireless cellular networks, increasing demands for higher speed data rates transmission, mobility, and multiservice access, have imposed staggering challenges. Therefore, IEEE 802.16 standards propose the use of Orthogonal Frequency Division Multiple Access (OFDMA) among multiple alternatives. OFDMA has become one of the most interesting developments in the area of new broadband wireless networks due to its powerful capability to mitigate Inter-Symbol Interference (ISI), provide high spectral efficiency and immunity of multipath fading.

Looking at wireless networks literature, several researches focus on adaptive resource allocation algorithms for single service required by users, in order to achieve some objectives aimed either to minimize total power under data rate constraint, called Margin Adaptive (MA) problem [1–3], or to maximize the total system throughput under power constraints referred as Rate Adaptive (RA) problem [4–6]. However, in practice how to efficiently allocate resources in multiservice wireless networks is not well-explored, nowadays. To resolve this challenge, recent multiservice transmission researches in wireless networks are paid more attention [7–9]. To handle a multiservice access network of heterogeneous traffic, the resource management scheme that can efficiently allocate subchannels to different users and services is essential.

The major characteristics of resource allocation algorithms consist of their running time, computational complexity, and efficiency. Generally, optimal resource allocation algorithms are classified as Nondeterministic Polynomial-time Hard (NP-Hard) problems, making them unsuitable for real-time applications such as video call. Therefore, literature tackles such problems by proposing suboptimal algorithms and heuristic methods in order to close optimal solution with low complexity and face real-time and channel variations constraints.

In this work, we propose two complementary methods in order to guarantee an efficient resource allocation policy. The first method gives new approach, in context of contiguous subchannel method, for subchannel gain computation using frequency responses dispersion. Our goal in this study is to

increase the number of bit per symbol subject to maintain lower BLoc Error Rate (BLER) in mobility and high-mobility context. The main objective in this second contribution is to resolve resource assignment to mobile users' problem, in order to take into account the trade-off between maximizing resources' use and fairness. In this context, a new dynamic heuristic algorithm is proposed. After that, we bring out multiservice feature of fourth generation wireless networks by proposing an adaptive resource allocation algorithm in OFDMA systems to support a variety of Quality-of-Service-(QoS-) sensitive applications such as streaming multimedia.

The remainder of this paper is organized as follows, In Section 2, system model is introduced and problem for resource allocation is formulated. In Section 3, a new method for subchannel gain is presented. Then, an adaptive subchannel allocation scheme is proposed. After that, a multi-QoS-based adaptive resource allocation algorithm is introduced in Section 5. Finally, numerical results are provided in Section 6.

## 2. System Model and Problem Formulation

In this work, we consider an OFDMA system for mobile wireless networks, based on IEEE 802.16 m standard. The system consists of a single Base Station (BS) that servers $K$ users and $N$ represents the number of subchannels composed by a group of $M$ adjacent subcarriers in a single subchannel with $N = L/M$ and $L$ is the total number of adjacent subcarriers. Considering $h_{k,n}$ as the channel gain of the $k$th user on $n$th subchannel, it following the complex Gaussian distribution and its magnitude, called fading factor, following Rayleigh distribution [10]. In fact, intercell interference occurs at a Mobile Station (MS), when the BSs of neighbouring cells transmit data over a subchannel used by its serving cell. The intercell interference phenomenon depends on user locations and mobilities, frequency reuse factor interfering cells as it is expressed by (2). We should notice that our considered cell in this work is not isolated. The downlink quality can be measured by the Signal-to-Interference plus Noise Ratio (SINR) and expressed as

$$\mathrm{SINR}_{e,k} = \frac{P_e}{(I_{e,k} + \Delta f N_0)} \left| h_{k,n} \right|^2, \tag{1}$$

where $P_e$ and $\Delta f$ are, respectively, the total transmit power and subchannel spacing. $N_0$ presents the Additive White Gaussian Noise (AWGN) variance. The average downlink interference per subchannel $I_{e,k}$ for the MS $k$ served by BS $e$ [11] is expressed as follows:

$$I_{e,k} = \sum_{s \neq e} \Lambda(e,s) \chi_s \left( \frac{P_s G_{s,k}}{L_{s,k}} \right), \tag{2}$$

where

(i) $P_s$ is the transmit power per subchannel of BS $s$;

(ii) $G_{s,k}$ is the antenna gain;

(iii) $L_{s,k}$ is the path-loss between BS $s$ and MS $k$;

(iv) $\chi_s$ is the probability that the same subchannel used by the mobile $k$ is used in the same time by another MS served by the BS $s$;

(v) $\Lambda(e,k)$ denotes the interference matrix, where the coefficient $\Lambda(e,k)$ equals 1 if cells $e$ and $s$ use the same band and zero otherwise.

The system capacity, $C_{\mathrm{syst}}$, is given by the following equation:

$$C_{\mathrm{syst}} = \sum_{t=1}^{T} \sum_{k=1}^{K} \sum_{n=1}^{N} \alpha_{k,n} C_{\mathrm{SC}_{t,k,n}}. \tag{3}$$

If the subchannel $n$ is allocated to user $k$ at the subframe $t$, $\alpha_{k,n}$ is equal to 1 and zero otherwise. The Parameter $C_{\mathrm{SC}_{t,k,n}}$ is the capacity of subchannel $n$ allocated to user $k$ at the subframe $t$ and is presented by

$$C_{\mathrm{SC}_{t,k,n}} = \sum_{m=1}^{M} C_{\mathrm{sbcr}_m}, \tag{4}$$

where $C_{\mathrm{sbcr}_m}$ is the transmitted information quantity of subcarrier in a subframe time. Assuming that all sub-carriers in each subchannel use the same AMC, (4) becomes

$$\begin{aligned} C_{\mathrm{SC}_{t,k,n}} &= M \cdot C_{\mathrm{sbcr}_m}, \\ C_{\mathrm{sbcr}_m} &= N_{\mathrm{AMC}_{k,n}} \cdot N_{\mathrm{sym/SF},} \end{aligned} \tag{5}$$

where $N_{\mathrm{AMC}_{k,n}}$ and $N_{\mathrm{sym/SF}}$ represent, respectively, the number of bits per symbol and the number of symbols per subframe. They depend on the modulation choice type.

Thus, the total system capacity is obtained as follows:

$$C_{\mathrm{syst}} = \sum_{t=1}^{T} \sum_{k=1}^{K} \sum_{n=1}^{N} \alpha_{k,n} \cdot M \cdot N_{\mathrm{AMC}_{k,n}} \cdot N_{\mathrm{sym/SF.}} \tag{6}$$

The Bit Error Rate (BER) for QPSK and M-QAM modulations are determined by formulas approximations presented in Table 1. In order to measure efficiency of the proposed method, we have introduced the Jain fairness index [13], given as

$$J_{\text{Fair Index}} = \frac{\left( \sum_{k=1}^{K} R_{k,\mathrm{mean}} \right)^2}{K \cdot \sum_{k=1}^{K} \left( R_{k,\mathrm{mean}} \right)^2}, \tag{7}$$

where $R_{k,\mathrm{mean}}$ is the mean data rate of user $k$ in a simulation time. This indicator is given by the following formula:

$$R_{k,\mathrm{mean}} = \frac{1}{b} \sum_{i=1}^{b=T/T_{\mathrm{SF}}} R_{k,i}, \tag{8}$$

where $R_{k,i}$ is the data rate of MS $k$ in subframe $i$.

In this work, equal power is allocated to subchannels in downlink sense in order to reduce computational complexity.

Table 1: Approximate BER for coherent modulations [12].

| Modulation | Formula |
|---|---|
| QPSK | $P_b \approx Q(\sqrt{2\gamma_b})$ |
| M-QAM | $P_b \approx \frac{4(\sqrt{M}-1)}{\sqrt{M}\log_2 M} Q\left(\sqrt{\frac{3\overline{\gamma}_b \log_2 M}{(M-1)}}\right)$ |

Having the target to maximize the system capacity, the objective function is formulated as follows:

$$\begin{aligned}
\text{Maximize} \quad & C_{\text{syst}} = \sum_{t=1}^{T}\sum_{k=1}^{K}\sum_{n=1}^{N} \alpha_{k,n} \cdot M \\
& \cdot N_{\text{AMC}_{k,n}} \cdot N_{\text{sym/SF}} \\
& J_{\text{Fair Index}} = \frac{\left(\sum_{k=1}^{K} R_{k,\text{mean}}\right)^2}{K \cdot \sum_{k=1}^{K} \left(R_{k,\text{mean}}\right)^2} \\
\text{Subject to} \quad & \text{C1: } \alpha_{k,n} \in \{0,1\}, \quad \forall k \in \Psi, n \in \Gamma \\
& \text{C2: } \sum_{k=1}^{K} \alpha_{k,n} = 1, \quad \forall n \in \Gamma \\
& \text{C3: } \sum_{n=1}^{N} \alpha_{k,n} \leq 1, \quad \forall k \in \Psi.
\end{aligned} \tag{9}$$

The two constraints C1 and C2 are on subchannel allocation to ensure that each subchannel is assigned to only one user where $\Psi$ and $\Gamma$ denote, respectively, the set of active users and subchannels in the cell. The constraint C3 denotes that one MS could have only one subchannel at the same time.

## 3. Efficiency of Sliding Window Gain Computation Method

In IEEE 802.16 m systems, the total sub-carriers of one bandwidth are grouped into subchannels in order to reduce the computational complexity and the signalling overhead [14–16]. Each subchannel is presented by a global channel gain that has a crucial role in the subchannels allocation policy in the case of multiuser wireless OFDMA systems.

In order to compute the global subchannel gain, different methods are described in [17–19]. In [18], a conservative estimation at the channel quality is made by choosing the most unfavorable channel of each group to represent that group's channel quality. Similarly, in [20, 21], the minimum channel gain approach is selected and considered as intuitive from an information-theoretic framework to ensure error-free transmission on all sub-carriers contained in the subchannel. However, if there are several sub-carriers with high channel gain, then the minimum sub-carriers gain method degrades the total system capacity and the maximum subchannel capacity is not being reached. In other works, the subchannel is denoted by an average channel gain for the corresponding set of sub-carriers. In [17], the channel magnitude response for each user is divided into a number of partitions where each partition is represented by the average channel gain. The same approach is adopted in [22] that allocates a subset of blocks with highest average channel gains to the corresponding user. In this case, sub-carriers with high channel gain are penalized by those with bad channel gain even though these sub-carriers represent a minority compared to those with high channel gain. Then, existing methods degrade the total system capacity.

In this section, we propose a new method to compute the subchannel's gain depending on frequency responses. The idea here is to close the channel quality based on dispersion probability and average channel gain. We can define the sub-carrier gain array as

$$\mathrm{H} = \begin{pmatrix} h_{1,1} & h_{1,2} & \cdots & h_{1,L} \\ h_{2,1} & h_{2,2} & \cdots & h_{2,L} \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ h_{K,1} & h_{K,2} & \cdots & h_{K,L} \end{pmatrix}, \tag{10}$$

where $K$ and $L$ represent, respectively, the number of users and the number of available sub-carriers in the system. $h_{k,l}$ is the channel gain of the $l$th sub-carrier for the $k$th user. We assume that a subchannel is composed by $M$ adjacent sub-carriers where $N = L/M$. When sub-carriers are grouped into subchannels, the subchannel gain array is obtained as

$$\mathrm{H}' = \begin{pmatrix} h'_{1,1} & h'_{1,2} & \cdots & h'_{1,N} \\ h'_{2,1} & h'_{2,2} & \cdots & h'_{2,N} \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ h'_{K,1} & h'_{K,2} & \cdots & h'_{K,N} \end{pmatrix}. \tag{11}$$

Our proposed "Sliding Window" method is a recursive scheme that depends on the average of sub-carriers gain and the dispersion probability. It gathers sub-carriers into three groups presented by a quality coefficient $Q$ that may vary from a type of service to another with $0.5 \leq Q \leq 1$.

(i) The number of sub-carriers with high channel gain is greater than that with bad channel gain: $P > Q$.

(ii) The number of sub-carriers with bad channel gain is greater than that with high channel gain: $P < (1 - Q)$.

(iii) The number of sub-carriers with bad and high gain is almost the same: $P \in [1 - Q, Q]$.

Our proposed method is described as in Algorithm 1 .

After computing the available subchannels gain, let us move to the subchannel allocation scheme in a single-service context.

## 4. A New Heuristic Method for Subchannel Allocation in Loaded System

In a loaded system, the number of users $K$ is greater than the number of available subchannels $N$. Here, the BS may assign to each user only one subchannel and a subchannel is allocated to only one user during a subframe. Our proposed resource allocation scheme consists of 3 steps: (1) *Initialization step*. Vectors and matrices that will be used later in the algorithm are initialised. (2) *Users ordering step*. Users are

**BEGIN**
**for** $k = 1$ to $K$ **do**
  **for** $n = 1$ to $N$ **do**
(i) ***Initialization***
$vect \leftarrow \{h_{k,1}, h_{k,2}, \ldots, h_{k,M}\}$; $\forall l \in \{1, 2, \ldots, L\}$;
$\forall m \in \{1, 2, \ldots, M\}$; $vect_{\max} \leftarrow \phi$; $vect_{\min} \leftarrow \phi$.
(ii) ***Subchannel Gain Computation***
**Step** (1):
Calculate the average channel gain $avg$ based on the sub-carriers frequency responses where $avg \leftarrow average(vect)$.
**Step** (2)**:**
Calculate the probability $p$ where $p \leftarrow \sum_{m=1}^{M} x_m/M$, with $\sum_{m=1}^{M} x_m \leq M$ and

$$x_m = \begin{cases} 1 & \text{if } h_{k,m} > avg, \\ 0 & \text{if not.} \end{cases}$$

**if** $p > Q$ **then**
  **for** $m = 1$ to $M$ **do**
    **if** $h_{k,m} > avg$ **then**
      $vect_{\max} = vect_{\max} + \{h_{k,m}\}$.
    **end if**
  **end for**
  **if** $length\ (vect_{\max}) = 1$ **then**
    **return** $gain \leftarrow vect_{\max}(1)$.
  **else**
    $vect \leftarrow vect_{\max}$; Jump to Step (1).
  **end if**
**end if**
**if** $p < (1 - Q)$ **then**
  **for** $m = 1$ to $M$ **do**
    **if** $h_{k,m} < avg$ **then**
      $vect_{\min} = vect_{\min} + \{h_{k,m}\}$.
    **end if**
  **end for**
  **if** $length(vect_{\min}) = 1$ **then**
    **return** $gain \leftarrow vect_{\min}(1)$.
  **else**
    $vect \leftarrow vect_{\min}$; Jump to Step (1)
  **end if**
**end if**
**if** $p \in [1 - Q, Q]$ **then**
  **for** $m = 1$ to $M$ **do**
    **if** $h_{k,m} < avg$ **then**
      $vect_{\min} = vect_{\min} + \{h_{k,m}\}$.
    **end if**
  **end for**
  **if** $length(vect_{\min}) = 1$ **then**
    **return** $gain_{\min} \leftarrow vect_{\min}(1)$.
  **else**
    $vect \leftarrow vect_{\min}$; Jump to Step (1)
  **end if**
  **for** $m = 1$ to $M$ **do**
    **if** $h_{k,m} > avg$ **then**
      $vect_{\max} = vect_{\max} + \{h_{k,m}\}$.
    **end if**
  **end for**
  **if** $length(vect_{\max}) = 1$ **then**
    **return** $gain_{\max} \leftarrow vect_{\max}(1)$.
  **else**

Algorithm 1: Continued.

```
                    vect ← vect_max; Jump to Step (1).
                end if
                return gain ← average(gain_max, gain_min).
            end if
            h'_{k,n} ← gain.
        end for
    end for
    END
```

Algorithm 1: Our sliding window scheme.

sorted in decreasing order depending on their best subchannel gain given by Algorithm 1. (3) *Subchannels allocation step*. Here, two cases are defined as it is depicted in Algorithm 2. Firstly, if only the selected user has this order, then we verify if its best subchannel is free. If yes, we allocate it and update the rate. If not, we search for the next free and best subchannel. Secondly, if two users or more have the same order, we verify if they require the same subchannel. If yes, we choose the user with the minimum second subchannel gain because it has a low chance to get a good subchannel (Algorithm 2).

*Computational Complexity.* Let us recall that $K$ refers to the users number and $N$ is the subchannels number. The users ordering step (2) sorts subchannels in descending order for each user. The sorting process requires $(N\log_2(N))$ operations for each user. Sorting subchannels in decreasing order for $K$ users requires then $KN\log_2(N)$. Thus, the asymptotic complexity is $O(KN\log_2(N))$. As proportional fairness method [23] includes a remaining subchannels allocation phase to ensure fairness criterion, computational complexity is equal to $O(KN\log_2(N)+K(N-K)\log_2(N-K))$. Alternative factor method [24] includes two steps: (1) Subchannels ordering step requires $KN\ \log_2(N)$ operations. (2) Users ordering step requires $K\log_2(K)$ operations. Alternative factor computation requires $(KN)$ operations. These steps pertain to subchannel allocation and the asymptotic complexity is equal to $O(KN\log_2(N) + K\log_2(K) + (KN))$. Then, we conclude that our suboptimal resource allocation scheme provides lower complexity than other existing methods and may be adopted for real-time applications. However, the key feature of 4G mobile network is its capability to support multiservice access when a user requires different service types, as it is well underlined in the next section.

## 5. Multi-QoS-Based Adaptive Resource Allocation Algorithm

We consider three Classes of Service (CoS) which are real-time Polling Service (rtPS), non-real-time Polling Service (nrtPS), and Best Effort (BE). For each CoS, the QoS satisfaction has a distinct definition [7]. The proposed algorithm consists of three steps: resource distribution step, calculation of each user's priority step, and resource allocation step.

*5.1. Resource Distribution.* We assume that $N_{\text{rtPs}}$, $N_{\text{nrtPs}}$, and $N_{\text{BE}}$ represent, respectively, the number of subchannels reserved to rtPS, nrtPS, and BE classes and are determined by the following equations: $N_{\text{rtPs}} = \alpha N$, $N_{\text{nrtPs}} = \beta N$, and $N_{\text{BE}} = \gamma N$, where $\alpha$, $\beta$ and $\gamma \in [0,1]$ and $\alpha + \beta + \gamma = 1$. Let $NC_{\text{rtPS}}$, $NC_{\text{nrtPS}}$, $NC_{BE}$ represent and, respectively, rtPS connections number, nrtPS connections number and BE connections number, and $NC$ denotes the total connections number. Proportional parameters are initially determined by $\alpha = (NC_{\text{rtPS}}/NC)$, $\beta = (NC_{\text{nrtPS}}/NC)$, and $\gamma = (NC_{\text{BE}}/NC)$. They are dynamically updated depending on the system availability. If necessary, a class, as rtPS-class, may reserve free subchannels from lower prior classes as nrtPS and BE classes and the opposite case is not allowed.

*5.2. Calculation of User's Priority.* For each user, a queue is used for buffering arrival packets in the proposed BS scheduler. We design the scheduling priority of each connection based on its channel quality, QoS satisfaction and service type priority.

*5.2.1. Real-Time Traffic.* For rtPS traffic of user $k$ on subchannel $n$, the scheduling priority $P_{k,n}$ is defined as [25]

$$P_{k,n} = \begin{cases} \beta_{\text{rtPS}}\left(\dfrac{R_{k,n}}{R_{\max}}\right)\left(\dfrac{1}{F_k}\right) & \text{if } F_k > 1, R_{k,n} \neq 0, \\ \beta_{\text{rtPS}}\left(\dfrac{R_{k,n}}{R_{\max}}\right) & \text{if } F_k \leq 1, R_{k,n} \neq 0, \\ 0 & \text{if } R_{k,n} = 0, \end{cases} \tag{12}$$

where $\beta_{\text{rtPS}} \in [0,1]$ is the rtPS class coefficient as defined in [25] and $F_k$ is the delay satisfaction indicator and is defined as

$$F_k = \frac{T_k - 2T_F}{W_k}, \tag{13}$$

where

(i) $R_{k,n}$ is the information bits that can be carried by user $k$ on the subchannel $n$ in one OFDM symbol,

(ii) $R_{\max}$ is the most bits per symbol that can be allocated when the most efficient AMC mode is selected,

(iii) $W_k$ is the longest packet waiting time of user $k$,

(iv) $T_k$ is the maximum packet latency of user $k$,

(v) $T_F$ is the frame duration.

**BEGIN**
(i) ***Initialization***
Equal power is allocated to groups
$\Gamma = \{1, 2, \ldots, N\}$; $\Psi = \{1, 2, \ldots, K\}$; $h_{k,n}$, $\forall k \in \Psi$, $\forall n \in \Gamma$; $\alpha_{k,n} = 0$, $\forall k \in \Psi$, $\forall n \in \Gamma$; $R_k = 0$, $\forall k \in \Psi$.
(ii) ***Users Ordering***
**for** $k = 1$ to $K$ **do**
  Sort $h_{k,n}$ in decreasing order.
  Order user $k$ according to its best channel gain $h_{k,1}$.
**end for**
(iii) ***Subchannels Allocation***
**for** $k = 1$ to $K$ where $order\ (k) \leq order\ (k+1)$ **do**
  $n \leftarrow$ find the subchannel $n$ where the user $k$ has its best channel gain.
  **Step** (1)**:**
  **if** $(uniqueorder = 1)$ **then**
    {% only user $k$ has this order}.
    *Sub-step* (1.*a*)*:*
    **if** $\alpha_{1 \to K,n} = 0$ **then**
      {% the subchannel $n$ is not allocated}
      $\alpha_{k,n} \leftarrow 1$ {% allocate the subchannel n}
      $\Gamma \leftarrow \Gamma - \{n\}$; Updated data rate $R_k$
    **end if**
    *Sub-step* (1.*b*)*:*
    **if** $\alpha_{1 \to K,n} = 1$ **then**
      {% the subchannel n is allocated}
      **repeat**
        $j = j + 1$ {% search the next free subchannel}
      **until** $\alpha_{1 \to K,j} = 0$ **or** $h_{k,j} < h_{k^*,j}$ {$k^*$ is the user less priority than the actual one and $k^* \in [k+1, K]$.}
      Jump to Sub-step (1.*a*)
    **end if**
  **end if**
  **Step** (2)**:**
  **if** $(uniqueorder = 0)$ **then**
    {% two or more users have the same order}.
    *Sub-step (2.a):*
    **if** $(uniquesubchannel = 0)$ **then**
      {% users with the same order do not require the same subchannel}
      jump to Sub-step (1.*a*) or Sub-step (1.*b*)
    **end if**
    *Sub-step* (2.*b*)*:*
    **if** $(uniquesubchannel = 1)$ **then**
      {% these users require the same subchannel}
      **if** $\alpha_{1 \to K,n} = 0$ **then**
        $\hat{k} \leftarrow$ determine the user that has the minimum second best sub-channe {% this user has a low chance to get a good subchannel}.
        Jump to Sub-step (1.*a*)
      **end if**
      **if** $\alpha_{1 \to K,n} = 1$**then**
        $\hat{k} \leftarrow$ determine the user that has the minimum second best subchannel; Jump to Sub-step (*1.b*)
      **end if**
    **end if**
  **end if**
**end for**
**END**

Algorithm 2: An adaptive resource allocation scheme.

We should notice that in this rtPS-class, the packet should be immediately sent if its deadline expires before the next frame is totally served. The ratio $(R_{k,n}/R_{\max})$ denotes the normalized channel quality. If $W_k \geq T_k - 2T_F$ we set the highest priority to the corresponding packet. When $R_{k,n} = 0$, the channel is in deep fade and this connection should not be served.

*5.2.2. Non-Real-Time Traffic.* For the nrtPS connection of user $k$ on subchannel $n$, the scheduling priority is defined as [25]

$$P_{k,n} = \begin{cases} \beta_{\text{nrtPS}}\left(\dfrac{R_{k,n}}{R_{\max}}\right)\left(\dfrac{1}{F_k}\right) & \text{if } F_k \geq 1, R_{k,n} \neq 0, \\ \beta_{\text{nrtPS}}\left(\dfrac{R_{k,n}}{R_{\max}}\right) & \text{if } F_k < 1, R_{k,n} \neq 0, \\ 0 & \text{if } R_{k,n} = 0, \end{cases} \tag{14}$$

where $\beta_{\text{nrtPs}} \in [0,1]$ is the nrtPS class coefficient as defined in [25] and $F_k$ is the ratio of the average transmission rate$\bar{r}_k$ over minimum reserved rate $r_k$, representing the rate satisfaction indicator. If $F_k \geq 1$ the rate requirement is satisfied. If not, representing the case within the queue will be full, packets of user $k$ should be then sent as soon as possible.

*5.2.3. Best Effort Traffic.* For the BE service of user $k$ on subchannel $n$, the scheduling priority is defined as [26]

$$P_{k,n} = \beta_{\text{BE}}\left(\frac{R_{k,n}}{R_{\max}}\right), \tag{15}$$

where $\beta_{\text{BE}} \in [0,1]$ is the BE-class coefficient as defined in [25]. In fact for the BE service, the scheduling priority depends only on the channel quality.

We should notice that the role of $\beta_{\text{rtPS}}$, $\beta_{\text{nrtPS}}$, and $\beta_{\text{BE}}$, is to provide different priorities for different QoS classes, then $\beta_{\text{rtPS}} > \beta_{\text{nrtPS}} > \beta_{\text{BE}}$ [25]. The purpose behind this idea is that the QoS of connections in a high-priority QoS class can be satisfied prior to those of low-priority QoS class.

After calculating the priority of each user $k$ on each subchannel $n$, we define a Priority Function that represents the user's priority and is described as

$$P_k = \max P_{k,n}, \quad \forall n \in \{1,\ldots,N\}. \tag{16}$$

*5.3. Resource Allocation Scheme.* We should notice that in this study, each user requires a single service at the same time and packets buffered in the same queue follow a First In First Out (FIFO) scheduler. Let $k^*$ denote the selected serving user, where

$$k^* = \max P_k, \quad \forall k \in \{1,\ldots,K\}. \tag{17}$$

After that, the user $k^*$ picks its best subchannel $n^*$ where

$$n^* = \max\left(\frac{R_{k^*,n}}{R_{\max}}\right), \quad \forall n \in \{1,\ldots,N\}. \tag{18}$$

The number of OFDMA symbols allocated for each user is calculated under the assumption that the minimum reserved rate $r_k$ and maximum latency $T_k$ should be satisfied for rtPS-class and only the minimum reserved rate $r_k$ for nrtPS-class. For each connection, a minimum data rate $r_k$ should be guaranteed and expressed by the following inequality:

$$r_k \leq \bar{r}_k(m+1), \tag{19}$$

where $\bar{r}_k(m+1)$ is the average service data rate of user $k$ at frame $(m+1)$. It is estimated over a windows size $T_c$ as [26]

$$\bar{r}_k(m+1) = \bar{r}_k(m)\left(1 - \frac{1}{T_c}\right) + \frac{\Delta_k(m+1)/\ T_F}{T_c}, \tag{20}$$

where $\Delta_k(m+1)$ is the number of information bits that should be sent during the frame $(m+1)$ of user $k$, where [26]

$$\Delta_k(m+1) = \begin{cases} (r_k - \bar{r}_k(m))\ T_c + \bar{r}_k(m)T_F & \text{if } \bar{r}_k(m) < r_k. \\ 0 & \text{if } \bar{r}_k(m) \geq r_k. \end{cases} \tag{21}$$

The number of slots required to carry $\Delta_k(m+1)$ information bits on subchannel $n$ equals

$$N_{\text{sub},k}(m+1) = \left\lceil \frac{\Delta_k(m+1)}{M \cdot R_{k,n}} \right\rceil. \tag{22}$$

For rtPS connection, there is one more step to calculate the number of information bit to be sent. We examine the waiting time of packets in the queue of connection from the head of line until the first packet whose waiting time is longer than $(T_k - 2T_F)$ is encountered. We denote $\Delta'_k(m+1)$ the sum of bits of packets from the head of line to the finding packet. The number of information bits allocated to rtPS connection of user $k$ is given by [26]

$$\hat{\Delta}_k(m+1) = \max\left\{\Delta_k(m+1), \Delta'_k(m+1)\right\}. \tag{23}$$

The number of slots required to carry $\hat{\Delta}_k(m+1)$ information bits on subchannel $n$ equals

$$N_{\text{sub},k}(m+1) = \left\lceil \frac{\hat{\Delta}_k(m+1)}{M \cdot R_{k,n}} \right\rceil. \tag{24}$$

If the available slot on subchannel $n$ is less than $N_{\text{sub},k}$, the second best subchannel for that user is selected and the remaining bits are allocated.

## 6. Simulation Results

In this work, the channel is modelled as a Rayleigh Channel with four multipaths. The simulated system consists of a single cell that uses 1024 sub-carriers for communications. In order to consider the mobility, the channel state changes every subframe delay and the simulation window is equal to 10000 subframes. Simulation parameters are described in Table 2.

Table 2: OFDMA parameters for IEEE 802.16 M.

| Parameter | Symbol | Value |
|---|---|---|
| Subchannels number | $N$ | 48 |
| Subcarriers number per subchannel | $M$ | 18 |
| Number of subframes per frame | | 7 |
| Sub-carriers spacing | $\Delta f$ | 7.813 KHz |
| Super frame delay | | 20 ms |
| Frame delay | $T_F$ | 5 ms |
| Subframe delay | | 714,286 $\mu$s |

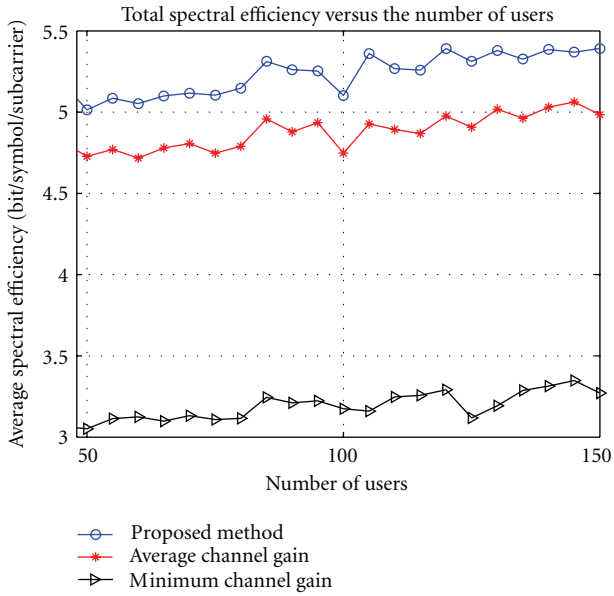



Figure 1: Total spectral efficiency versus the number of users in loaded systems.

Table 3: Variation intervals in terms of total spectral efficiency.

| | [48, 75[ | [75, 100[ | [100, 125[ | [125, 150] |
|---|---|---|---|---|
| $\text{TSEVI}_{\text{Pag}}$ (bit/sym) | 0.197 | 0.193 | 0.250 | 0.246 |
| $\text{TSEVI}_{\text{Pmi}}$ (bit/sym) | 1.823 | 1.883 | 1.899 | 1.923 |

*6.1. Proposed Sliding Window Method Performances.* We show the performance of our sliding window method compared to the minimum [18] and average [19] method.

Figure 1 compares the average spectral efficiency per subcarrier versus the number of users. The total spectral efficiency is determined by (6).

Table 3 shows variation intervals in terms of total spectral efficiency. Let $\text{TSEVI}_{\text{Pag}}$ and $\text{TSEVI}_{\text{Pmi}}$ denote the total spectral efficiency for different variation user intervals. These values are computed based on, respectively, the mean difference between sliding window and average channel gain method and the mean difference between sliding window and minimum channel gain method. As $\text{TSEVI}_{\text{Pag}} > 0$ and $\text{TSEVI}_{\text{Pmi}} > 0$, for all intervals, it is obvious that the proposed method provides greater spectral efficiency, because it computes a subchannel gain that closes the channel quality.

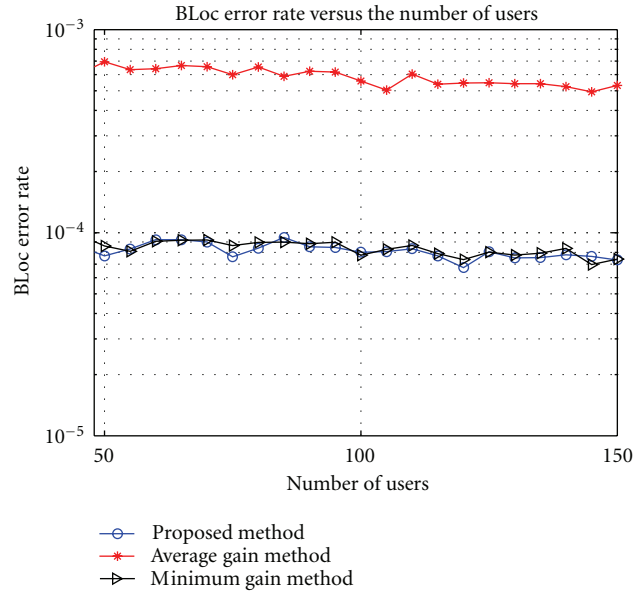



Figure 2: BLoc Error Rate (BLER) versus the number of users.

Table 4: Variation intervals in terms of BLER.

| | [48, 75[ | [75, 100[ | [100, 125[ | [125, 150] |
|---|---|---|---|---|
| $\text{BVI}_{\text{Pag}}10^{-4}$ (dB) | −5.718 | −5.320 | −4.724 | −4.535 |
| $\text{BVI}_{\text{Pmi}}10^{-6}$ (dB) | −1.371 | −3.870 | −2.233 | −0.977 |

Figure 2 illustrate the average Bloc Error Rate (BLER) versus the number of users.

Table 4 shows variation intervals in terms of BLER. Let $\text{BVI}_{\text{Pag}}$ and $\text{BVI}_{\text{Pmi}}$ denote the average BLER for different variation user intervals. These values are computed based on, respectively, the mean difference between sliding window and average channel gain method and the mean difference between sliding window and minimum channel gain method. As $\text{BVI}_{\text{Pag}} < 0$ and $\text{BVI}_{\text{Pmi}} \leq 0$, for all intervals, our proposed method provides lower BLER than the average gain method and quasisimilar BLER compared to minimum gain method, because our scheme closes the channel quality.

*6.2. Proposed Subchannels Allocation Algorithm Performances in a Single Service Context.* Our proposed resource allocation algorithm is compared with the suboptimal existing solutions [23, 24]. The reason for this comparison is as follows. Shen et al. [23] formulates the problem of maximizing the total system capacity with proportional rate constraints. It uses the subchannel with high SINR for each user. Hwang et al. [24] proposes a heuristic for channel allocation. In this work, an alternative factor is defined for subchannel allocation. It aims to increase the downlink system capacity while maintaining sufficient fairness.

Figure 3 compares the average spectral efficiency per subcarrier versus the number of users.

Table 5 shows variation intervals in terms of total spectral efficiency. Let $\text{SEVI}_{\text{PAF}}$ and $\text{SEVI}_{\text{PPF}}$ denote the average spectral efficiency in different variation user intervals. These

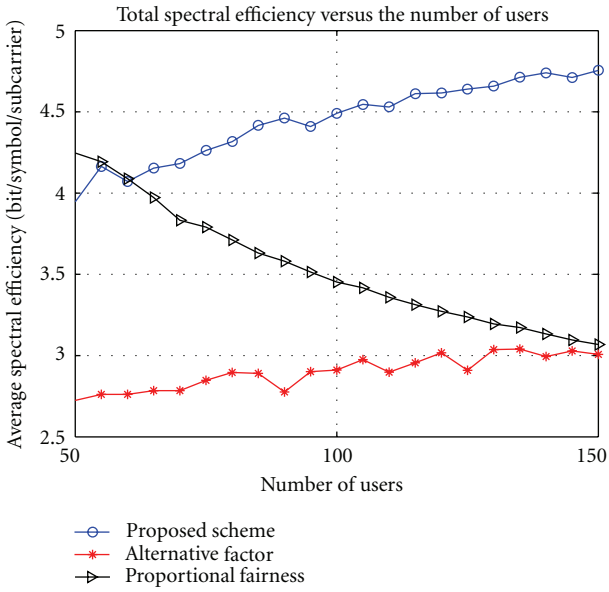



Figure 3: Total spectral efficiency in loaded systems.

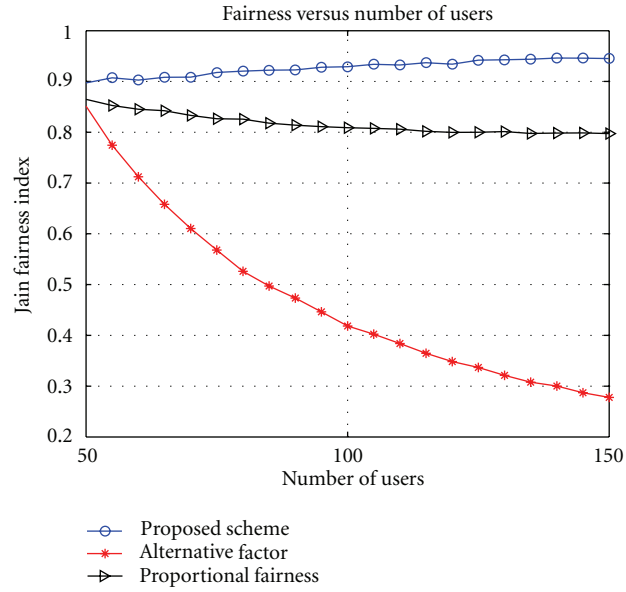



Figure 4: Jain Fairness Index versus the number of users in loaded systems.

Table 5: Variation intervals in terms of total spectral efficiency.

| | [50, 60] | ]60, 75[ | [75, 100[ | [100, 125[ | [125, 150] |
|---|---|---|---|---|---|
| $SEVI_{PAF}$ (bit/s) | 1.31 | 1.38 | 1.51 | 1.60 | 1.70 |
| $SEVI_{PFF}$ (bit/s) | −0.11 | 0.26 | 0.72 | 1.19 | 1.55 |

Table 6: Variation intervals in terms of Jain Fairness Index.

| | [50, 75[ | [75, 100[ | [100, 125[ | [125, 150] |
|---|---|---|---|---|
| $JFIVI_{PAF}$ | 0.183 | 0.420 | 0.549 | 0.639 |
| $JFIVI_{PPF}$ | 0.057 | 0.103 | 0.128 | 0.145 |

values are computed based on, respectively, the mean difference between proposed scheme and alternative factor method [24] and the mean difference between proposed scheme and proportional fairness method [23]. As $SEVI_{PAF} > 0$ and $SEVI_{PPF} > 0$, when the number of users $K \in ]60,150]$, our proposed method provides greater spectral efficiency, because it covers the loaded system case when the number of available resources is lower than the number of users requiring access to the cell. However, when $K \in [50, 60]$, the proposed scheme provides lower SE than the proportional fairness method as $SEVI_{PPF} < 0$. Hence, the contribution of our proposed scheme performs better when the number of users in the cell is important which is close to the practical case.

To better examine the fairness of these algorithms for different number of users, their performance is shown in Figure 4. The Jain Fairness Index is expressed by (7). It is obvious that the proposed method provides a fairness index close to 1.

Table 6 shows variation intervals in terms of average Jain Fairness Index. Let $JFIVI_{PAF}$ and $JFIVI_{PPF}$ denote the Jain Fairness Index for different variation user intervals. These values are computed based on, respectively, the mean

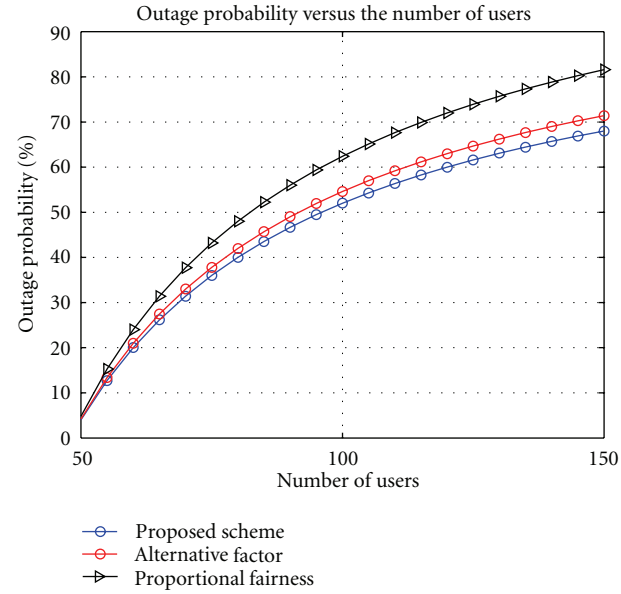



Figure 5: Outage probability versus the number of users in loaded systems.

difference between proposed scheme and alternative factor method described in [24] and the mean difference between proposed scheme and proportional fairness method proposed in [23]. As $JFIVI_{PAF} > 0$ and $JFIVI_{PPF} > 0$, for all intervals, it is obvious that the proposed method provides more fairness among active users than the alternative factor and proportional fairness method as each user may reserve only a single subchannel at any given time as it is expressed by constraint C3 in our optimisation problem formulation.

Figure 5 shows the outage probability versus the number of users where the outage probability represents the rejected

Table 7: Variation intervals in terms of outage probability.

| | [50, 75[ | [75, 100[ | [100, 125[ | [125, 150] |
|---|---|---|---|---|
| $OPVI_{PAF}$ (%) | −0.943 | −2.156 | −2.809 | −3.247 |
| $OPVI_{PPF}$ (%) | −3.772 | −8.626 | −11.236 | −12.991 |

Table 8: Variation intervals in terms of total spectral efficiency in unloaded systems.

| | [10, 12] | ]12, 15[ | [15, 20[ | [20, 25[ | [25, 30] |
|---|---|---|---|---|---|
| $USEVI_{PMM}$ (bit/s) | −0.231 | 0.237 | 0.687 | 1.123 | 1.458 |
| $USEVI_{PPF}$ (bit/s) | −0.087 | 0.265 | 0.709 | 1.132 | 1.466 |

users percentage. When a user does not get its required subchannel, the number of rejected users increases by one. Then, the outage probability is the ratio of the number of rejected users and the total number of active users in a loaded system.

Table 7 shows variation intervals in terms of outage probability. Let $OPVI_{PAF}$ and $OPVI_{PPF}$ denote the Outage probability for different users variation intervals. These values are computed based on, respectively, the average of difference between proposed scheme and alternative factor method described in [24] and the average of difference between proposed scheme and proportional fairness method proposed in [23]. As $OPVI_{PAF} < 0$ and $OPVI_{PPF} < 0$, for all intervals, it is obvious that the proposed scheme provides lower outage probability than other methods. We should notice that in this case, as it is shown by Table 7, the difference between the proposed heuristic and existing methods [23, 24] increases when the number of users rises, meaning that the proposed scheme satisfies a greater number of users than other existing methods [23, 24] in a loaded system case, because it covers the inefficiency resource case by allocating to each user its second best and free subchannel.

For unloaded system case, the performance of our proposed algorithm is compared with the algorithms proposed in [23, 27]. The reason for this comparison is as follows. Authors in [23] aim to maximize the total system capacity while maintaining proportional fairness among active users. The principle of this suboptimal subchannel algorithm is to use the subchannels with high SINR for each user. For remaining subchannels, the user with the lowest proportional capacity has the option to pick which subchannel to use in order to achieve proportional fairness. Resource allocation algorithm proposed in [27] aims to assign to each user a subchannel in which the user has the best channel conditions. For the remaining subchannels, the user with the lowest amount of capacity is selected and its best subchannel is allocated to him in order to reach a sufficient fairness among users.

Figure 6 compares the average spectral efficiency per subcarrier versus the number of users.

Table 8 shows variation intervals in terms of total spectral efficiency for proposed. Let $USEVI_{PMM}$ and $USEVI_{PPF}$ denote the average total spectral efficiency for different users variation intervals in unloaded systems. These values are computed based on, respectively, the mean difference between proposed scheme and max-min method described in [27]

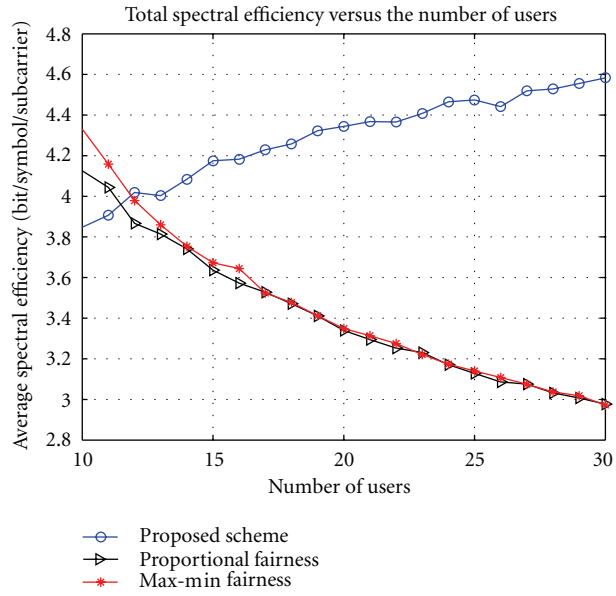



Figure 6: Total spectral efficiency versus the number of users in unloaded systems.

and the mean difference between proposed scheme and proportional fairness method proposed in [23]. As $USEVI_{PMM} > 0$ and $USEVI_{PPF} > 0$, when $K \in ]12,30]$, it is obvious that the proposed method provides greater spectral efficiency than the max-min and proportional fairness method. We should notice that in this case, as it is shown by Table 8, the difference between the proposed method and existing methods [23, 27] increases when the number of users rises. Then, the proposed method operates well the multiusers diversity. However, when $K \leq 12$, the existing method in [23, 27] provides better performance in terms of total spectral efficiency as $USEVI_{PMM} < 0$ and $USEVI_{PPF} < 0$ when $K \in [10, 12]$, because they introduce a remaining subchannel allocation phase. So, one user may have more than one subchannel as it is described above. Hence, the contribution of our proposed scheme performs better when the number of users in the cell is important which is generally close to the practical case when good subchannels are not yet available.

*6.3. Proposed Subchannels Allocation Scheme Performances in a Heterogeneous Traffics System.* The performance of the Multi-QoS-based adaptive resource allocation proposed algorithm is compared to two existing algorithms that are proposed in [28, 29] in terms of rtPS average Packet Loss Rate (PLR)and nrtPS Packet Satisfaction Ratio (PSR). On one hand, Maximum-Carrier-to-Interference and Noise Ratio (MAX-CINR) scheme [28] allocates resources to the user with the maximum receiver CINR and then only the users' link qualities are concerned and the QoS requirements are totally ignored. On the other hand, Modified Proportional Fair (MPF) scheduling algorithm proposed in [29] is taken into account with QoS guaranteed to users in the system. In order to evaluate the performance of various QoS services, we use the Near Real Time Video (NRTV) traffic model for

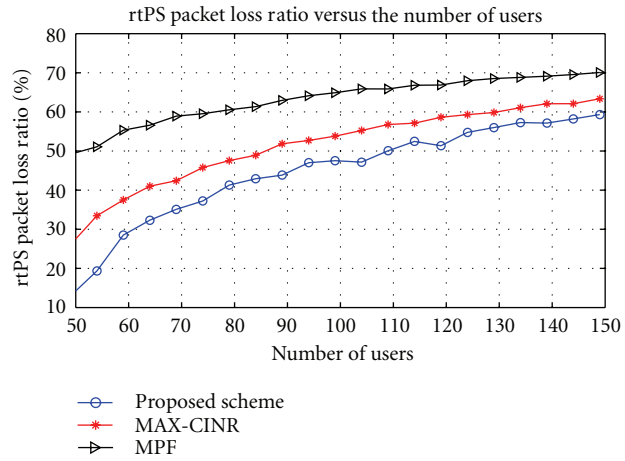



Figure 7: Average rtPS packet loss ratio versus the number of users.

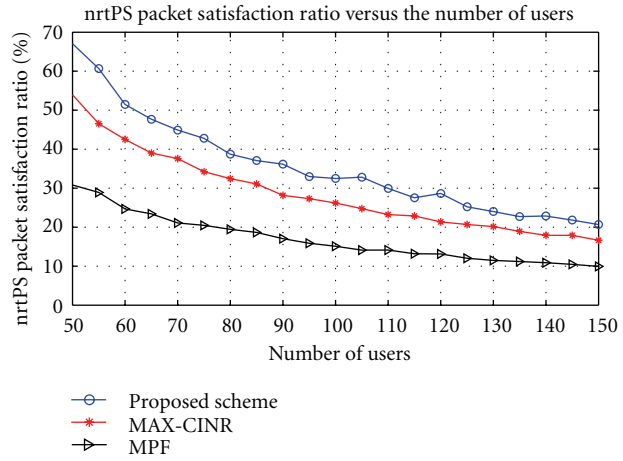



Figure 8: Average nrtPS packet satisfaction ratio versus the number of users.

Table 9: Variation intervals in terms of rTPS packet loss ratio.

| | [50, 75[ | [75, 100[ | [100, 125[ | [125, 150] |
|---|---|---|---|---|
| $HPLRVI_{PMCI}(\%)$ | −20.88 | −13.78 | −9.204 | −8.351 |
| $HPLRVI_{PMPF}(\%)$ | −24.84 | −16.72 | −14.21 | −10.33 |

Table 10: Variation intervals in terms of nrTPS packet satisfaction ratio.

| | [50, 75[ | [75, 100[ | [100, 125[ | [125, 150] |
|---|---|---|---|---|
| $HPSRVI_{PMCI}$ (%) | 9.078 | 5.994 | 5.740 | 3.631 |
| $HPSRVI_{PMPF}$ (%) | 27.21 | 18.32 | 15.57 | 11.32 |

rtPS service, FTP model for nrtPS service [30]. Here we heuristically define length of rtPS packets 1024 bits, nrtPS 2048 bits, and BE 4096 bits and we set the QoS class coefficients as $\beta_{rtPS} = 1.0$, $\beta_{nrtPS} = 0.8$ and $\beta_{BE} = 0.6$. Moreover, we assume that packet arrival process is Poisson distributed, each connection with its own average arrival rate. We suppose that there are 150 users in the system where each one requires a single service at the same time. For rtPS connection, the minimum reserved rate and maximum latency of each connection are set to 500 kbps and 20 ms, respectively. For nrtPS connection, the minimum reserved rate is set as 1 Mbps. For BE connection, the buffer size is 5000 packets with 512 bytes each.

Figure 7 shows the average Packet Loss Ratio (PLR) of the rtPS connection across different number of users. The average PLR is defined as the ratio of the number of the lost rtPS packets to the total packets' number.

Table 9 shows variation intervals in terms of rtPS Packet Loss Ratio. Let $HPLRVI_{PMCI}$ and $HPLRVI_{PMPF}$ denote the PLR in different variation user intervals in heterogeneous services system. These values are computed based on, respectively, the average of difference between proposed scheme and MAX-CINR method described in [28] and the average of difference between proposed scheme and modified proportional fairness method proposed in [29]. As $HPLRVI_{PMCI} < 0$ and $HPLRVI_{PMPF} < 0$, for all intervals, it is obvious that the proposed method satisfies more rtPS users than the MAX-CINR and modified proportional fairness method, because our scheduler gives the priority to rtPS-class to ensure an adequate resource allocation.

In Figure 8, we investigate the average nrtPS Packet Satisfaction Ratio (PSR) which is defined as the ratio of the number of the connections guaranteeing the minimum reserved rate to the total connections number. Table 10 shows variation intervals in terms of nrtPS Packet Satisfaction Ratio. Let $HPSRVI_{PMCI}$ and $HPSRVI_{PMPF}$ denote the PSR in different variation user intervals in heterogeneous services systems. These values are computed based on, respectively, the mean difference between proposed scheme and MAX-CINR method [28] and the mean difference between proposed scheme and modified proportional fairness method [29]. As $HPSRVI_{PMCI} > 0$ and $HPSRVI_{PMPF} > 0$, for all intervals, it is obvious that the proposed method satisfies more nrtPS users than the MAX-CINR and MPF method, due to our proposed reservation phase that reserves subchannels to rtPS, nrtPS, and BE services according to proportional parameters depending on system resources availability. For other methods, there are no resource reservation phase, which can increase the nrtPS calls rejection.

*6.4. Concluding Remarks.* Simulation results demonstrate that our sliding window method increases the system capacity and decreases the BLER effectively compared the minimum and the average channel gain methods. In loaded systems, simulation results show that the proposed algorithm permits to achieve a better trade-off between fairness and efficiency use of resources compared to other recent methods [23, 24]. Moreover, our proposed resource allocation scheme proves efficient in unloaded system than other existing methods. In addition to this contribution, the new heuristic algorithm present a low complexity and may be adopted for real-time and mobile applications. In a heterogeneous services context, simulation results illustrate that our proposed scheme can simultaneously satisfy the QoS requirements of various classes services: rtPS, nrtPS, and BE.

## 7. Conclusion

In order to reduce the OFDMA system complexity, available sub-carriers are grouped into equal groups of contiguous sub-carriers, where each group is called a subchannel. The adaptive modulation and coding scheme, AMC, is used in order to maximize the number of bit per symbol. In this paper, we have firstly proposed a new method for subchannel gain computation based on the frequency responses dispersion. Secondly, we have proposed a new heuristic method for subchannels allocation problem in the context of WiMAX release 2.0, IEEE 802.16 m. An adaptive method for subchannels allocation was necessary in order to exploit the multi-user diversity, to respect real-time constraints and to maximize the system capacity. The idea of this method was based on the statistic parameters, mean, variance, root mean square, or RMS of the frequency response channel gain for every mobile station. Finally, we proposed a multi-QoS-based resource allocation algorithm for OFDMA systems. We defined a priority function for each user according to the QoS satisfaction degree and its corresponding subchannel qualities. Simulation results showed that proposed algorithms provide a better trade-off between total system capacity, fairness, and complexity compared to other existing methods. For future work, we are interested to validate the present proposition in a multiservice context in order to develop an efficient radio resources management policy for Long-Term Evolution (LTE) network.

## References


[1] C. Y. Wong, R. S. Cheng, K. B. Letaief, and R. D. Murch, "Multiuser OFDM with adaptive subcarrier, bit, and power allocation," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 10, pp. 1747–1758, 1999.

[2] W. Rhee and J. M. Cioffi, "Increase in capacity of multiuser ofdm system using dynamic subchannel allocation," in *Proceedings of the 51st Vehicular Technology Conference (VTC '00)*, vol. 2, pp. 1085–1089, 2000.

[3] W. Yu and R. Lui, "Dual methods for non-convex spectrum optimization of multi-carrier systems," *IEEE Transactions on Communications*, vol. 54, no. 7, pp. 1310–1322, 2006.

[4] J. Jang and K. B. Lee, "Transmit power adaptation for multiuser OFDM systems," *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 2, pp. 171–178, 2003.

[5] M. Fathi and H. Taheri, "Utility-based resource allocation in orthogonal frequency division multiple access networks," *IET Communications*, vol. 4, no. 12, pp. 1463–1470, 2010.

[6] H. Zhu and J. Wang, "Adaptive chunk-based allocation in multiuser ofdm systems," in *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC '10)*, pp. 1525–3511, 2010.

[7] W. Lihua, M. Wenchao, and G. Zihua, "A cross-layer packet scheduling and subchannel allocation scheme in 802.16e ofdma system," in *Proceedings of the Wireless Communications and Networking Conference (WCNC '07)*, pp. 1865–1870, 2007.

[8] K. Juhee, K. Eunkyung, and S. K. Kyung, "A new efficient B S scheduler and scheduling algorithm in WiBro systems," in *Proceedings of the 8th International Conference Advanced Communication Technology (ICACT '06)*, pp. 1467–1470, 2006.

[9] Y. F. Yu, H. Ji, and G. X. Yue, "A novel wireless IP packets scheduling algorithm with QoS guaranteeing based on OFDM system," *Journal of Beijing University of Posts and Telecommunications*, vol. 29, no. 3, pp. 61–65, 2006.

[10] Z. Huiling and J. Wang, "Adaptive chunk-based allocation in multiuser OFDM systems," in *Proceedings of the Wireless Communications and Networking Conference (WCNC '10)*, pp. 1–6, IEEE, 2010.

[11] R. Nasri and Z. Altman, "Handover adaptation for dynamic load balancing in 3gpp long term evolution systems," in *Proceedings of the MoMM*, pp. 145–154, 2007.

[12] A. Goldsmith, *Wireless Communications*, vol. 572, Cambridge University Press, 2005.

[13] R. K. Jain, D. W. Chiu, and W. R. Hawe, "A quantitative measure of fairness and discrimination for resource allocation in shared computer system," Tech. Rep. 301, DEC, 1984.

[14] D. Marabissi, D. Tarchi, R. Fantacci, and A. Biagioni, "Adaptive subcarrier allocation algorithms in wireless OFDMA systems," in *Proceedings of the IEEE International Conference on Communications (ICC '08)*, pp. 3475–3479, Beijing, China, May 2008.

[15] A. Gotsis, D. Komnakos, and P. Constantinou, "Dynamic subchannel and slot allocation for OFDMA networks supporting mixed traffic: upper bound and a heuristic algorithm," *IEEE Communications Letters*, vol. 13, no. 8, pp. 576–578, 2009.

[16] S. H. Paik, S. Kim, and H. B. Park, "A resource allocation using game theory adopting AMC scheme in multi-cell OFDMA system," in *Proceedings of the 2nd International Conference on Future Computer and Communication (ICFCC '10)*, vol. 2, pp. 344–347, 2010.

[17] T. C. H. Alen, A. S. Madhukumar, and F. Chin, "Capacity enhancement of a multi-user OFDM system using dynamic frequency allocation," *IEEE Transactions on Broadcasting*, vol. 49, no. 4, pp. 344–353, 2003.

[18] Q. Wang, D. Xu, J. Xu, and Z. Bu, "A grouped and proportional-fair subcarrier allocation scheme for multiuser OFDM systems," in *Proceedings of the 25th IEEE International Performance, Computing, and Communications Conference (IPCCC '06)*, pp. 97–101, April 2006.

[19] U. Kim, H. Nam, and B. F. Womack, "An adaptive grouped-subcarrier allocation algorithm using comparative superiority," in *Proceedings of the Military Communications Conference (MILCOM '05)*, vol. 2, pp. 963–969, October 2005.

[20] R. Agarwal, R. Vannithamby, and J. M. Cioffi, "Optimal allocation of feedback bits for downlink OFDMA systems," in *Proceedings of the IEEE International Symposium on Information Theory*, pp. 1686–1690, 2008.

[21] J. Chen, R. A. Berry, and M. L. Honig, "Performance of limited feedback schemes for downlink OFDMA with finite coherence time," in *Proceedings of the IEEE International Symposium on Information Theory*, pp. 2751–2755, 2007.

[22] L. Xiaowen and Z. Jinkang, "An adaptive subcarrier allocation algorithm for multiuser OFDM system," in *Proceedings of the IEEE 58th Vehicular Technology Conference (VTC '03)*, pp. 1502–1506, October 2003.

[23] Z. Shen, J. G. Andrews, and B. L. Evans, "Adaptive resource allocation in multiuser OFDM systems with proportional rate constraints," *IEEE Transactions on Wireless Communications*, vol. 4, no. 6, pp. 2726–2736, 2005.

[24] S. Hwang, J. Park, Y. S. Jang, and H. S. Cho, "A heuristic method for channel allocation and scheduling in an OFDMA system," *ETRI Journal*, vol. 30, no. 5, pp. 741–743, 2008.

[25] L. Qingwen, W. Xin, and G. B. Giannakis, "A cross-layer scheduling algorithm with QoS support in wireless networks," *IEEE Transactions on Vehicular Technology*, vol. 55, no. 3, pp. 839–847, 2006.

[26] Z. Xinning, H. Jiachuan, Z. Song, Z. Zhimin, and D. Wei, "An adaptive resource allocation scheme in OFDMA based multiservice WiMAX systems," in *Proceedings of the 10th International Conference on Advanced Communication Technology*, pp. 593–597, February 2008.

[27] D. Marabissi, D. Tarchi, R. Fantacci, and A. Biagioni, "Adaptive subcarrier allocation algorithms in wireless OFDMA systems," in *Proceedings of the IEEE International Conference on Communications*, pp. 3475–3479, 2008.

[28] S. Ryu, B. Ryu, H. Seo, and M. Shin, "Urgency and efficiency based packet scheduling algorithm for OFDMA wireless system," in *Proceedings of the IEEE International Conference on Communications (ICC '05)*, pp. 2779–2785, May 2005.

[29] T.-D. Nguyen and Y. Han, "A dynamic channel assignment algorithm for OFDMA systems," in *Proceedings of the IEEE Vehicular Technology Conference (VTC '06)*, pp. 1273–1277, 2006.

[30] cdma2000 evaluation methodology, 3gpp2 contribution c.r1002-0. 1, December 2004.

Hindawi Publishing Corporation
International Journal of Digital Multimedia Broadcasting
Volume 2012, Article ID 189325, 11 pages
doi:10.1155/2012/189325

*Research Article*

# Performance Evaluation of an Object Management Policy Approach for P2P Networks

**Dario Vieira,[1] Cesar A. V. Melo,[2] and Yacine Ghamri-Doudane[3,4]**

[1] *LRIE Lab, École d'Ingénieur des Technologies de l'Information et de la Communication (EFREI), 94800 Villejuif, France*
[2] *Department of Computing Science, Federal University of Amazonas, 69077-000 Manaus, AM, Brazil*
[3] *École Nationale Supérieure d'Informatique pour l'Industrie et l'Entreprise (ENSIIE), 91025 Evry, France*
[4] *LIGM Lab, Université Paris-Est, 75420 Champs-sur-Marne, France*

Correspondence should be addressed to Dario Vieira, dario.vieira@efrei.fr

Received 1 June 2011; Revised 21 September 2011; Accepted 5 October 2011

Academic Editor: Ivan Lee



The increasing popularity of network-based multimedia applications poses many challenges for content providers to supply efficient and scalable services. Peer-to-peer (P2P) systems have been shown to be a promising approach to provide large-scale video services over the Internet since, by nature, these systems show high scalability and robustness. In this paper, we propose and analyze an object management policy approach for video web cache in a P2P context, taking advantage of object's metadata, for example, video popularity, and object's encoding techniques, for example, scalable video coding (SVC). We carry out trace-driven simulations so as to evaluate the performance of our approach and compare it against traditional object management policy approaches. In addition, we study as well the impact of churn on our approach and on other object management policies that implement different caching strategies. A YouTube video collection which records over 1.6 million video's log was used in our experimental studies. The experiment results have showed that our proposed approach can improve the performance of the cache substantially. Moreover, we have found that neither the simply enlargement of peers' storage capacity nor a zero replicating strategy is effective actions to improve performance of an object management policy.

## 1. Introduction

The increasing popularity of network-based multimedia applications poses many challenges for content providers to supply efficient and scalable services. Peer-to-peer (P2P) systems have been shown to be a promising approach to provide large-scale video services over the Internet since, by nature, these systems show high scalability and robustness.

One of the essential problems in measuring how these systems perform is the analysis of their properties in the presence of churn, a collective effect created by the independent arrival and departure of peers. Resiliency, key design parameters, and content availability are issues in P2P systems that are influenced by the churn. Hence, the user-driven dynamics of peer participation must be taken into account in both design and evaluation of any P2P system and its related mechanics, such as object management policies. For instance, when peers go off-line, the locally stored data become unavailable as well. This temporary content unavailability can lead to congestion on backbone links due to the extra traffic generated by requesting to the original content provider.

The main contributions of this paper are as follows.

(1) First, we propose and analyze an object management policy approach for videos' web cache in a P2P context. In this approach we exploit the object's metadata, for example, video popularity, and object's encoding techniques, for example, scalable video coding (SVC). In Section 5, we describe the object management policy based on popularity (POP); that is, we describe how user-generated content is used to define this object management policy. We have carried out set of studies by using three different scenarios so as to analyze our approach with regarding other object management policies. We evaluated how the insertion, replacement, and discarding of videos impacts

the object management policies. Besides, we study as well the effects of the video popularity and the discarding of video layers in the performance of object management policies. In Section 6 we present the numerical results collected from these simulation experiments. We show as well how much our content-oriented web cache mechanism can improve traditional web cache mechanisms. Basically, we have found that our approach outperforms the traditional one and consequently reduce the capacity demand poses over the community output link by increasing the hit rate of community demands and optimizing the network resources available.

(2) Second, we study the impact of the user churn on object management policies. For that, we have used a churn model that is based on a stationary alternating renewal process with its average up and down behavior based on the type of peer (cf. Section 4). By using this churn model, we have evaluated the enlargement of individual peer storage capacity as an action to keep policies performance since the community storage capacity will be affected directly by the churn. In Section 7, we present the numerical results collected from these simulation experiments. We can point that peers will be able to store valuable objects over a long time-scale which could improve content availability. We have noted that the gap among all police performance shrinks due to this enlargement. Nonetheless, this enlargement does not impact linearly the performance of the policies.

(3) Finally, we have evaluated on how much the replicated data affects policies performance by measuring it on a system with and without duplicated data. Indeed, each time a peer leaves the system, it will make chunks of popular content unavailable for other peers. Whenever that peer rejoins the system, its content could have been accessed by other peer. Accordingly, data are naturally replicated into the system due to the churn. This naturally replication has as consequence the decreases of the nominal system storage capacity. So, we have evaluated on how much this replicated data affects policies performance by measuring it on a system with and without duplicated data. In Section 7.1, we present the numerical results collected from these simulation experiments. We have found that the performance of policies is impacted, either positively or negatively, by this replicated data which suggests that content availability could be improved whether the volume of duplicated data is under policy control.

We have used in our experimental studies a YouTube video collection, which records over 1.6 million video's log. In our experiments, each peer can store up to 25 short videos of 4:50 minutes—the average video length identified in the studied video collection. The overall storage capacity, which is defined by the sum of the storage capacity of all peers, is equal to one percent of the video collection storage demand (cf. Section 3). Different simulated scenarios were defined by scaling up the system storage capacity until 20% of the video collection storage demand.

We have evaluated five policies in our experiments: (i) the context-aware and content-oriented policy (POP), (ii) least recently used (LRU) policy, (iii) least frequently used (LFU) policy, (iv) popularity-aware greedy-dual size (GDSP) policy [1], and (v) proportional partial caching (PARTIAL) policy [2]. The last four policies (described in Section 2) are representative implementations of their classes, that is, recency-based, frequency-based, and cost-based policies.

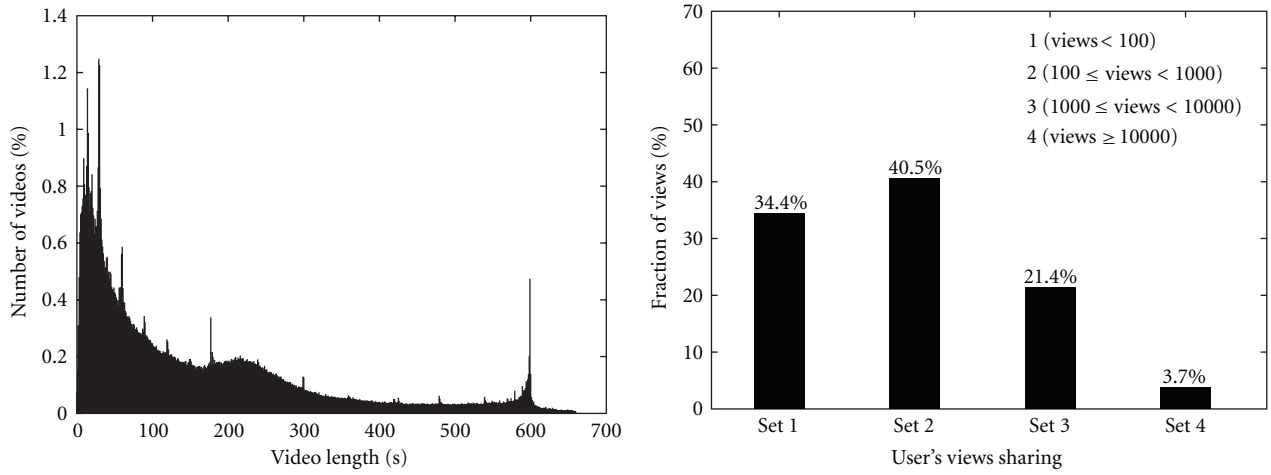
## 2. Cache Algorithms

This section gives a short overview of the traditional cache algorithms that we use through this paper. Essentially, these algorithms can be classified into three major flavors of strategies as follows.

*Recency-Based Strategies.* This kind of approach uses recency as a main factor. Least recently used (LRU) strategy, and all its extensions, is a good example of these strategies. LRU makes use of the locality information to predict future accesses to objects from past accesses. In effect, there are two sorts of locality: (i) temporal locality and (ii) spatial locality. The first one is based on the idea that recently accessed objects are likely to be used again in the future. In the latter approach, the references to some objects suggest accesses to other objects. Recency-based strategies make use of temporal locality; that is, LRU exploits temporal locality. The major disadvantage of these strategies is that they do not consider object size information. Moreover, they do not consider frequency information.

*Frequency-Based Strategies.* These strategies exploit the frequency as a main factor. They are based on the idea that different objects have different popularity values and this implies that these objects have different frequency values. Accordingly, these values are used for future decisions. The least-frequently used (LFU) is the well-known implementation of these strategies. There are two kinds of implementation of LFU (and its extensions): (i) perfect LFU, which counts all requests to an object. This keeps on across replacement; (ii) in-cache LFU, whose counts are carried out only for cache objects. It should be noted that this does not represent all request in the past. The major disadvantage is that the frequency counts are too static for dynamic environments.

*Cost-Based Strategies.* In cost-based policies, the importance of an object is defined by a value got from a cost function. When a cache runs out of space, objects are evicted based on their cost; objects with the smallest cost will be evicted first. The greedy-dual size (GDS) policy was the first policy to implement a cost-based cache strategy. It maintains, for each object, a characteristic value Hi which is set in the very first request and calculated every time the object is requested. Improvements on GDS policy have been implemented to




(a) Videos length (b) Popularity

Figure 1: Statistics for YouTube Entertainment category [3].

consider historical and access frequency data (GDSP—popularity-aware greedy-dual size [1]) and the dynamics of peer-to-peer system (proportional partial caching) [2].

## 3. Video Collection

In this section, we present the video collection used so as to carry out our experiments. Our dataset consists of meta-information about user-generated videos from YouTube services. We limited our experiments to the *Entertainment* category collected by [3] owing to the massive scale of YouTube. This collection consists of 1,687,506 videos where each line represents a single video. Furthermore, each video record contains both fixed and time-varying information (e.g., length, views, ratings, stars, and URL), which means

(1) views and ratings stand for the number of times the video has been played and evaluated by users,

(2) stars stand for the average score from rating, and

(3) URL denotes the list of external web pages hyperlinking the video.

We limited the maximum size of cacheable video to 99 minutes; videos with more than this value are considered crawler mistake. Figure 1(a) depicts videos' length distribution in seconds, with video length being equal to 291 seconds.

Figure 1(b) shows the video popularity distribution. By examining the number of requests recorded in our video collection, we grouped those videos in four sets. In the first set we gathered video with less than 100 views. This subset is made of 34.4% of videos in our collection. In the second subset we gathered videos with a number of views into the range of 100 views and 1000 views, which contains 40.5% of videos in our collection. The third subset has videos with a number of views into the range of 1,000 views and 10,000 views and makes up 21.4% of videos recorded in our collection. Finally, in the fourth subset we gathered videos that recorded over 10.000 views, which contains 3.7% of videos in our collection. Based on our analysis, we see that video requests are highly concentrated on a small set of videos. In fact, the fourth subset has 60% of the total number of views.

This video collection misses individual user requests information; that is, timestamps mark that record when a user dispatched his/her requests. To deploy our simulation studies, we must have such individual user's behavior. Hence, we developed a procedure to simulate the users' behavior based on information gathered from the video collection. Therefore, in order to use this collection, we define a procedure for the generation of requests, as follows.

(1) All requests will be driven by a combination of the video length and the "think time."

(2) For each two nonpopular videos, we picked up three popular videos. This ratio was derived based on the preview analysis carried over our video collection; that is, we determined the number of (non)popular video views recorded.

(3) The pick-up procedure of popular and not popular videos is independent and follows a uniform distribution.

(4) A popular video must be recorded more than 10,000 views.

The rationales behind our procedure are (i) downloaded videos will be watched end-to-end, and users will spend some time looking for related videos, the thinking time, before dispatching a new request; (ii) over the time scale the system is observed, popular and nonpopular videos will keep their status, a reasonable assumption since we are interested only on the effectiveness of popularity as a criterion to manage those videos; (iii) in our collection, videos with more than 10,000 views represent less than 4% of the whole collection but recorded over 60% of the total number of views.

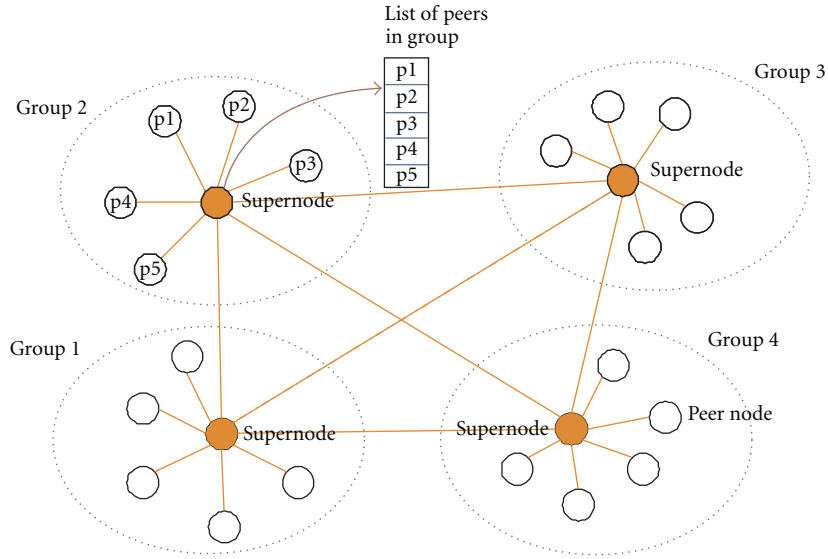



Figure 2: Two-tier hierarchy framework.

## 4. A Hierarchical Content-Oriented Overlay Network

In this section, we present the networking scenario used in our studies. First, the network infrastructure is considered, then we describe how peers are classified and objects are searched in this infrastructure. Finally, the churn model used to characterize peers' dynamics is presented.

*4.1. The Network Infrastructure.* The network infrastructure is a structured peer-to-peer (P2P) system made by ordinary nodes and supernodes and with peers that are locality-aware clustered into communities or groups. These communities are built around subjects, for example, science, sports, and DIY. Figure 2 illustrates this infrastructure. Peers locality awareness limits content lookups to very few hops into the community. In Crespo and Garcia-Molina [4], similar ideas have been proposed to share music files with overlays being built based on music genres, and lookup operations being bounded to the community.

In this network infrastructure, for each group there is one or more superpeers, which are analogous to gateways routers in hierarchical IP networks. The goal is to use superpeers so as to transmit messages intergroups.

On this networking scenario, the mix of gathered peers defines a community as homogeneous or heterogeneous. Accordingly, peers can be classified based on their up probability and they can be clustered in $\mathcal{T} > 1$ groups. In this paper, we classify and cluster them in four groups (i.e., $\mathcal{T} = 4$) as follows.

(i) *Stable group*, each peer is up with probability big or equal to $p$.

(ii) *Moderate group*, each peer is up with probability $t$, but with $p \gg t$.

(iii) *Low group*, each peer is up with probability $r$, but with $t \gg r$.

(iv) *Unstable group*, each peer is up with probability less or equal to $q$, but with $r \gg q$.

These peer group's up probabilities are mapped to the peer group's up session duration in our numerical studies as suggested by Wang et al. [5]. Specifically, we studied a community made by peers that fits in a low group behavior; that is, peers up sessions last 28 minutes in average, and a typical YouTube session duration by the time our video collection was collected, according to Wang et al. [5].

*4.2. The Churn Model.* To model the peers' dynamics, we have exploited a generic churn model defined by Yao et al. [6]. Consider a P2P system with $n$ peers, where each peer $i$ is either UP at time $t$ or DOWN. This behavior can be modeled by a renewal process $\{Z_i(t)\}$ for each peer $i$:

$$Z_i(t) = \begin{cases} 1, & \text{peer } i \text{ is alive at time } t, \\ 0, & \text{otherwise,} \end{cases} \quad 1 \le i \le n. \tag{1}$$

Unlike [6], the UP and DOWN lasting sessions of $\{Z_i(t)\}$ are based on the type of peer $i$. Therefore, the actual pairs $(F_i(x), G_i(x))$ are chosen randomly from set $\mathcal{F}$ define as follows.

$$\mathcal{F} = \left\{ \left(F^{(1)}(x), G^{(1)}(x)\right), \ldots, \left(F^{(\mathcal{T})}(x), G^{(\mathcal{T})}(x)\right) \right\}, \tag{2}$$

where $\mathcal{T} \ge 1$ is the number of peer types in the system.

Therefore, for each process $\{Z_i(t)\}$, its UP lasting sessions $\{L_{i,c}\}_{c=1}^{\infty}$ have some joint distribution $F_i(x)$, and its DOWN lasting sessions $\{D_{i,c}\}_{c=1}^{\infty}$ have another joint distribution $G_i(x)$, where $c$ stands for cycle number and durations of users and $i$'s UP and DOWN sessions are given by random variables $L_{i,c} > 0$ and $D_{i,c} > 0$, respectively.

Examples of UP/DOWN distributions used throughout this paper are (i) the exponential, defined as

$$F_i' = 1 - e^{-\lambda_i x}, \tag{3}$$

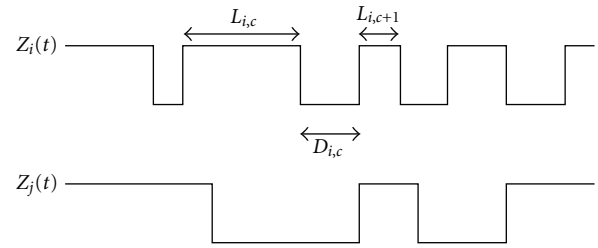
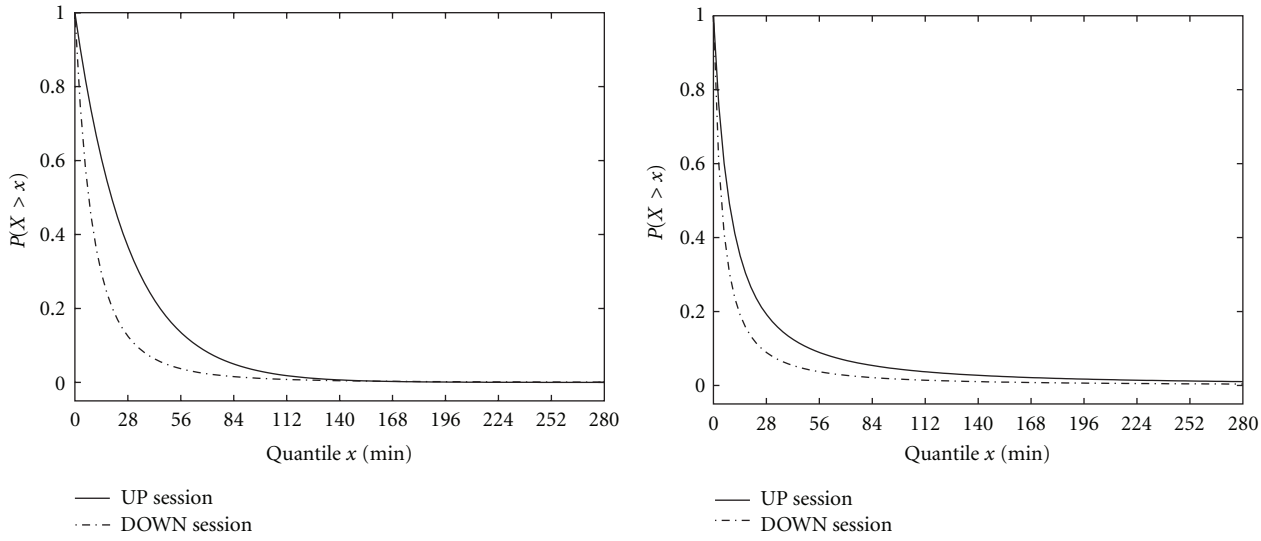



FIGURE 3: Up and down behaviors for peer $p_i$ and $p_j$. Processes $\{Z_i(t)\}$ and $\{Z_j(t)\}$ are independent.




(a) Exponential churn model

(b) Heavy tail churn model

FIGURE 4: Complementary cumulative distribution functions of UP and DOWN session duration used on exponential and heavy tail churn models.

with mean $1/\lambda_i$, and (ii) the Pareto, defined as

$$F_i^{''} = 1 - \left(1 + \frac{x}{\beta_i}\right)^{\alpha i}, \quad x > 0,\ \alpha_i > 1, \tag{4}$$

with mean $\beta_i/(\alpha_i - 1)$.

Based on the aforementioned model, the following assumptions are made.

(1) To capture the independent nature of peers, we assume that the set $\{Z_i(t)\}_{i=1}^{n}$ consists of mutually independent alternating renewal processes, as depicted in the Figure 3. Accordingly, peers behave independently of each other and processes $\{Z_i(t)\}$ and $\{Z_j(t)\}$, for any $i \neq j$, are independent.

(2) Note that $l_i$, the average UP session duration, and $d_i$, the average DOWN session duration, are independent and unique for each peer. Once pair $(l_i; d_i)$ is generated for each peer $p_i$, it remains constant for the entire evolution of the system.

(3) 28-minute YouTube average session duration is set to $l_i$. In addition, half of the value of $l_i$ is set to $d_i$. Hence, as regards content availability, a demanding community of peers is expected.

(4) The upprobability of peer $i$ at an arbitrary time instance $t$ is given by

$$p = \lim_{t \to \infty} P(\{Z_i(t)\} = 1) = \frac{l_i}{l_i + d_i}. \tag{5}$$

Based on previous characterization, two churn models have been used on our studies: heavy tail and exponential. When exponential churn model is applied, the lasting of UP session is driven by $\exp(1/l_i)$ distribution, while the lasting of DOWN session is driven by a *pareto* $(3, d_i)$ distribution, where $l_i$ stands for the expected mean of UP session duration and $d_i$ stands for the expected mean DOWN session duration.

Figure 4(a) shows the complementary cumulative distribution function (CCDF) of the exponential (UP session) and Pareto (DOWN session) distributions when $l_i$ and $d_i$ are set as mentioned previously. For exponential distribution, the probability of chose values greater than the expected ones vanishes after minute 120. However, for Pareto distribution, this probability still exists over large time-scale. This churn pattern can be summarized by the following: peers will stay connected, as they join the community, but will lose interest over time.

```
Require: video_popularity ∨ cache_size ≥ 0
    request_from_community(video);
    if miss_video then
        request_from_outside(video);
    end if
    if video_popularity > threshold then
        insert_cache(video);
    end if
```

Algorithm 1: The popularity-based object management policy.

In the heavy tail churn model, the UP and DOWN sessions are driven by the Pareto distributions. Different from exponential churn model, heavy tail churn model draws probability still significant over time, Figure 4(b), which means that picking a value great than the mean value is highly possible. The churn pattern defined by this heavy tail model can be summarized by the following: peers will keep interest on content over long time-scales.

## 5. POP: A Content-Oriented Management Policy Based on Popularity

As mentioned in Section 4.1, peers will cooperate and the interesting-oriented community has very useful information to improve the cache system performance. In this context, the probability that a video comes to be accessed again by other members of a community is higher than that in a general case since members inside a community might share common interests. Hence, the question about how much popular a video is inside a community seems to be an important metadata to be considered when implementing an object management policy.

Based on the aforementioned assumptions, we have developed an object management policy [7], that is, the popularity-based (POP) policy. In this object management policy, we keep the most popular video in cache (number of visualizations) based on a predefined threshold. The rationale of our policy is that the majority of video requests is targeted to the popular ones, hence keeping the cache filled by the popular video will probably improve the hit rate and decrease the volume of traffic that flows outside the community link. Algorithm 1 describes this procedure.

As said, the threshold used to identify popular videos is a predefined value and closed related to video collection statistics. In our studies we set the threshold equal to 10,000 visualizations which defines a popular video collection with less than 4% of the whole video collection, at the same time, this popular video collection has over 60% of all visualizations. In summary, the threshold setting procedure will reduce the number of videos that have to be managers, but those videos will receive the majority of requests.

## 6. Performance Analysis of POP

In this section we show the numerical results collected from our first studies. We evaluated how the insertion, replacement, and discarding of videos impact the object management policies. To evaluate the effectiveness of our proposed policy, we defined a reference scenario and we compared it to three other different scenarios so as to evaluate the performance of the POP. For that, we assume that peers are always connected to the system, a borrowed concept from [5] which identifies and describes the importance of stable peers.

```
Require: video_popularity ∨ cache_size ≥ 0
    if video_popularity > threshold then
        insert_cache(video);
    end if
```

Algorithm 2: Considering video popularity.

*6.1. Numerical Results.* We use a trace-driven simulation to evaluate our proposed policy. The simulated P2P network has 10,000 peers, and we assume that there is only one community where each peer can connect to any other peer. Each peer has a cache with capacity to storage up to 1.000 seconds of video. The total number of video requests is 200,000, and we consider that videos with more than a *threshold value* are popular. In our experiments, the threshold value is equal to 10,000 views.

*6.2. Reference Scenario.* In this scenario, we establish the following conditions: (i) every requested (and not yet cached) video must be added to the cache, (ii) LRU is the object replacement policy implemented by the web cache, and (iii) the whole video will be discarded when the managed cache is full. Taking into consideration these assumptions, we simulated the user requests based on a uniform distribution. As we have pointed out, the results are then compared with three other scenarios, which are described as follows.

*6.3. Scenario Number 1.* In this scenario, we evaluated the effects of the video popularity on the performance of our web cache. Instead of adding all referenced, and not yet cached videos, we keep only cached videos that are considered popular, that is, videos that have a number of views greater than 10,000. The rationale of our policy is that the majority of video requests is targeted to the popular ones, hence keeping the cache filled by the popular video will probably improve the hit rate and decrease the volume of traffic that flow outside the community link. In summary, in scenario number 1 we cached only popular videos, the object management policy is still the LRU, and the whole video is discarded when the managed cache is full.

Algorithm 2 describes the patched applied to the object management policy; that is, the execution of function *insert_cache(·)* is conditioned to the video popularity. Figures 5 and 6, the first bars, show the improvements, in terms of hit rate (35.2%) and volume of traffic that is saved (36.2%), when that new policy is implemented.

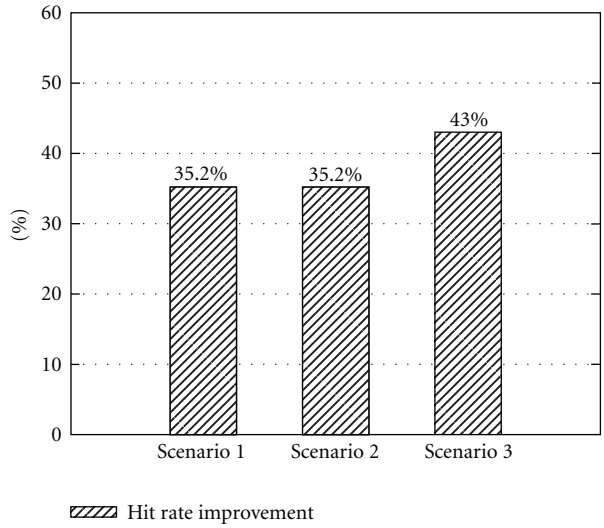



Figure 5: Hit rate—based on reference scenario.

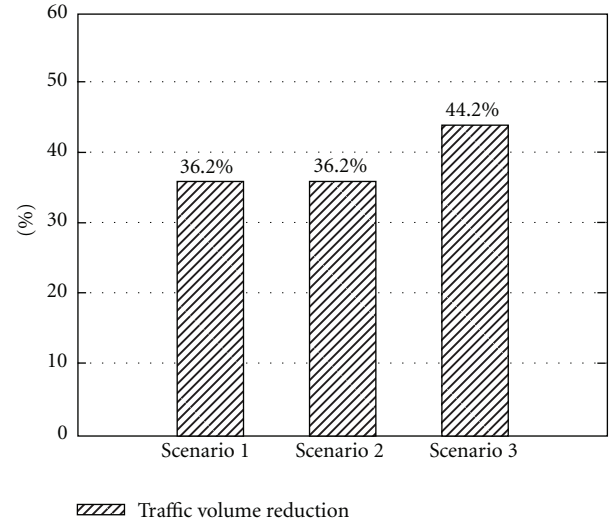



Figure 6: Traffic volume reduction—based on reference scenario.

```
Require: free_cache_size ∨ video_size
  While free_cache_size < video_size do
    (video,node) ⇐ LRU_video_object();
    nlayers ⇐ discard_highest_layer(video, node);
    if nlayers = 0 then
      nlayers ⇐ discard_highest_layer(video, node);
      remove_reference_DHT(video, node);
    end if
  end while
```

Algorithm 3: Discarding layers, the local approach.

```
Require: free_cache_size ∨ video_size
  While free_cache_size < video_size do
    (video, node) ⇐ random_video_object();
    nlayers ⇐ discard_highest_layer(video, node);
    if nlayers = 0 then
      nlayers ⇐ discard_highest_layer(video, node);
      remove_reference_DHT(video, node);
    end if
  end while
```

Algorithm 4: Discarding layers, the global approach.

*6.4. Scenario Number 2.* In this scenario, we evaluated the effects of the video popularity and the discarding of video layers on the performance of our object management policy. Differently to the scenario number 1, we dispose only video layers that are coding to improve the video quality and hence keep those layers that have the minimum amount of data to the receiver reproduce his/her requested video. The least recently used criterion is applied to elect which video must have its layers discarded. Whether the room made available by the discarded layer is still insufficient to store the recently accessed video, a new video is elected and its layer is discarded. This process is repeated until the web cache has sufficient space to store the recently referenced video. The rationale of our policy is to keep as much as possible a popular video cached, even it is a low-quality copy, since references to it are very probable.

Algorithm 3 presents the patch applied to object management policy. Function *LRU_video_object(·)* finds the least recently used cached object and returns both that video and the node that has the object. Function *discard_highest_layer (video, node)* discards the highest layer of the found video. For example, if there are two layers for this video in cache, the second, which has less priority, will be discarded. Additionally, after calling the *discard_highest_layer(·)* function no layers could remain for that video, hence the reference for this video must be removed from distributed hash table (DHT). Figures 5 and 6, the second bars, show the improvements in term of hit rate (35.2%) and saved traffic (36.2%) when a new policy is implemented.

*6.5. Scenario Number 3.* In this scenario, we introduce the layer discarding policy associated with a global approach instead of the local one deployed in scenario number 2. In other words, we randomly selected objects in cache and discarded the highest layer of this object. The rationale of our approach is that when a cache run out of space, we have a set of high popular videos cached with all those video showing a high requesting rate. Hence when we apply the discarding policy over the whole set of videos, and not just over the "least" recently used set, we will increase the set of popular videos that can be cached. Consequently, we can observe an improvement over the hit rate. In summary, scenario number 3 has only popular video cached, a random object replacement policy, and video layers are discarded when cache is full.

Algorithm 4 presents the patches applied to the object management policy. Function *random_video_object(·)* picks a cached object and returns both the targeted video and node that has that object.

Figures 5 and 6, the third bars, show the improvements in terms of hit rate (43%) and saved traffic (44.2%) when the new policy is implemented.

*6.6. Discussions.* Requests to popular video are much more probe than nonpopular videos; over 60% of requests recorded in our video collection are targeted to popular videos. In addition, popular videos made just 3.7% of our video collection. Based on these facts, we tested the video popularity metadata as the criteria to cache or dispose recently accessed videos. As noted from scenario number 1, there was a 30% improvement in the cache hit rate, compared to the traditional approach, reassuring that bringing metadata associated to the objects will save the network resources with server communities that were built around subjects.

Although 30% improvement is remarkable, we learnt that traditional web cache policies do not assume anything about the managed objects. In the scenario number 3, we exploit the fact that videos are distributed in layers. Besides, we use a discarding policy based on those layers instead of discarding the whole video, as implemented on traditional web cache policy. We underline that the discarding of layers does not constrain a web cache to server requests to that video; that is, videos can be served, for instance, in a low definition instead of high definition. As a result, whenever a cache runs out of space and a new object needs to be cached, the number of cached videos increase since the discarding of a whole video will be postponed, at least in the first moment.

An important finding to be explained is how LRU policy and video layers discarding policy are related, as pointed in scenario number 2. In this scenario, we discard video layers that belong to the set of the least recently used videos. Nonetheless, this situation does not offer any improvement in both the number and quality of cached objects. This happens because the group of least used objects is very small (only popular video are cached), which heavily restrict the space where our discarding policy can act to improve the number of cached videos.

We noted as well that bringing video metadata (i.e., video popularity) into an object management policy associated with a global layers discard policy can increase substantially the quality and number of cached videos. Consequently, this improves the cache hit rate and saved traffic measured over the output link. Using these approaches, we got over 40% improvement which could mean to postpone updates over an output link.

## 7. Numerical Results

In this section, we have carried out some studies so as to evaluate the impact of the churn on object management policies implemented by peers. Accordingly, we have set up a peer-to-peer-assisted video distribution system, compare Section 4, and we have measured the decreasing ratio, that is, on how much the policy performance measured by the hit rate is impacted by the churn. The decreasing ratio is defined by the following equation:

$$\mathrm{HR}_{\mathrm{Decr}} = 1 - \frac{\mathrm{HR}_{\mathrm{With}}}{\mathrm{HR}_{\mathrm{Without}}}, \tag{6}$$

where $\mathrm{HR}_{\mathrm{With}}$ and $\mathrm{HR}_{\mathrm{Without}}$ are, respectively, the measured hit rates in a system with and without churn. In our experiments, we have employed both exponential and heavy tail churn models (cf. Section 4.2).

As we have pointed out, the accomplishment of an object management policy is affected by the performance metrics taken into account. For instance, the LRU policy can have high hit rate but performs poorly in term of byte hit rate. Therefore, we have studied also the decreasing ratio defined by the byte hit rate collected in a system with and without churn. From a qualitative stand point of view, the impacts measured by the decreasing ratio defined by both, hit rate and byte hit rate, are similar. Hence, we show as well the decreasing ratio defined by (6).

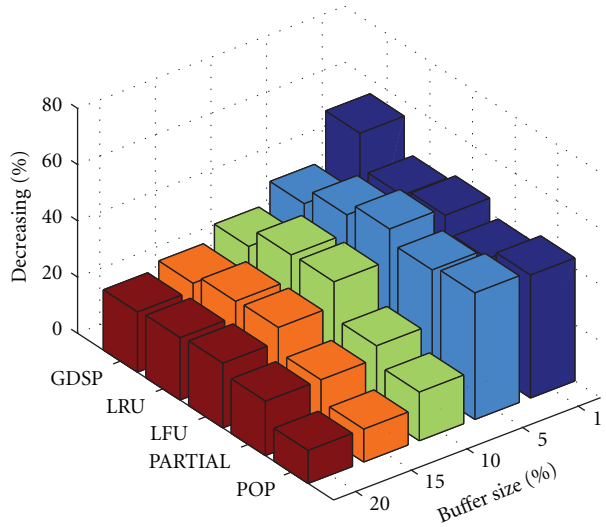
We have evaluated five policies: the context-aware and content-oriented policy (POP) [7], least recently used (LRU) policy, least frequently used (LFU) policy, popularity-aware greedy-dual size (GDSP) policy [1], and proportional partial caching (PARTIAL) policy [2]. The last four policies are representative implementations of their classes, that is, recency-based, frequency-based, and cost-based policies.

In our experiments, each peer can store up to 25 short videos of 4:50 minutes—the average video length identified in the studied video collection. The overall storage capacity, which is defined by the sum of the storage capacity of all peers, is equal to one percent of the video collection storage demand, see Section 3. We have defined different simulated scenarios by scaling up the system storage capacity until 20% of the video collection storage demand.

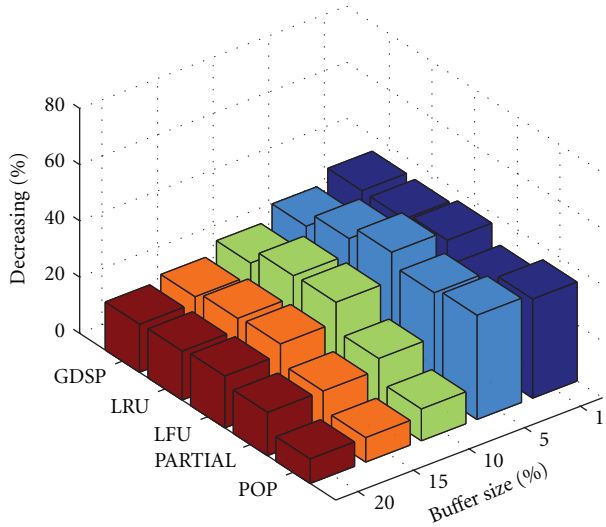
Figure 7 depicts the impact of the churn in term of decreasing on hit rate, when both exponential and heavy tail models drive the churn. Although their decreasing ratios, defined by (7), have different values, all policies' behavior is as follows: the gap among their performance shrinks as the system storage capacity is enlarged. However, this enlargement does not impact linearly the performance of policies. Specifically, for an enlargement of 20% in the system storage capacity, LFU policy has recorded a variation of 36% (29%) on its decreasing ratio for the exponential (heavy tail) churn model. LRU shows a variation of 34% (27%), while GDSP shows a variation of 33% (24%) and PARTIAL shows variation of 29% (23%).

The performance of POP policy represents an exception in the previous conclusion, specially for heavy tail churn model where the variation on its decreasing ratio is equal to 20%. That is, for the POP policy, the impact of churn is linearly reduced by the system enlargement. The volume of replicated data is the main reason for that performance (cf. Section 7.1).

The enlargement of the system storage capacity impacts policies performance, especially for exponential churn model where variations in the decreasing ratio are more pronounced as this enlargement happens. Figure 8 depicts the decreasing of system size, that is, the shrinking on the number of UP peers, when exponential and heavy tail churn models drive the joining and leaving events. When the churn is driven by the heavy tail model, the maximum decreasing on system size is 39%, whereas for exponential model, this value is 48%. As a general system design observation, we




(a) Exponential churn model




(b) Heavy tail churn model

FIGURE 7: Decreasing ratio of object management policies (see (6)).

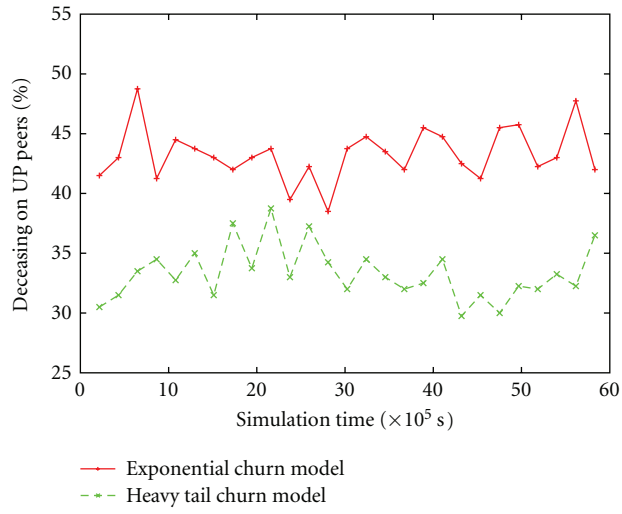



FIGURE 8: The decreasing on number of UP peers.

have noted that, since strong assumption on peers' storage capacity could jeopardize the system implementation, P2P-assisted systems have to deploy mechanisms to keep, over large time-scales, peers interested on content made available by communities.

*7.1. Replicated Data.* Each time a peer leaves the system, it will make chunks of popular content unavailable for other peers. Whenever that peer rejoins the system, its content could have been accessed by an other peer. This dynamic replicates data over peers that share interest and, consequently, decreases the nominal system storage capacity. Figure 9 shows the percentage of data that has been replicated into the system due to the churn.

For LRU, LFU, and GDSP policies, the maximum amount of replicated data demands 20% of storage capacity. However, for the two other policies, PARTIAL and POP, the amount of replicated data is around 25% and over 45%, respectively. The POP policy can handle twice as much replicated data than other policies. Since the POP policy keeps only objects with certain number of views (10,000 views in our experiments) and this requirement is too restrictive, the amount of replicated data grows rapidly due to the churn.

Resilience to failure is a key property of P2P systems. In this context, every time a peer $A$ fails in delivering a service, which is under its responsibility, another peer will replace the peer $A$ so as to deliver this service. This property has consequences in the proposed video distribution system; that is, the studied policies have to handle replicated data held independently by peers. From the preview results, at least 20% of stored data is replicated ones, this value being greater than 45% when the POP policy is used.

To measure the impact of replicating on policy performance, we have performed experiments by using the following nonreplicating procedure: peers must evict any replicated video every time they rejoin the system.

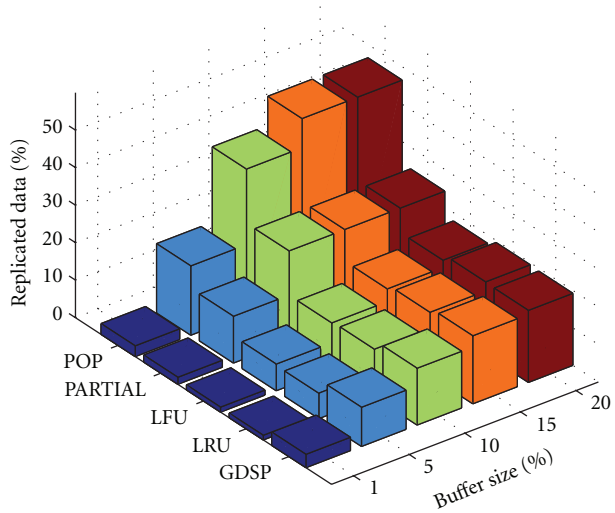
In this new scenario, we have computed the decreasing ratios as follows:

$$\mathrm{HR}_{\mathrm{DecrNorep}} = 1 - \frac{\mathrm{HR}_{\mathrm{NorepWith}}}{\mathrm{HR}_{\mathrm{Without}}}, \tag{7}$$

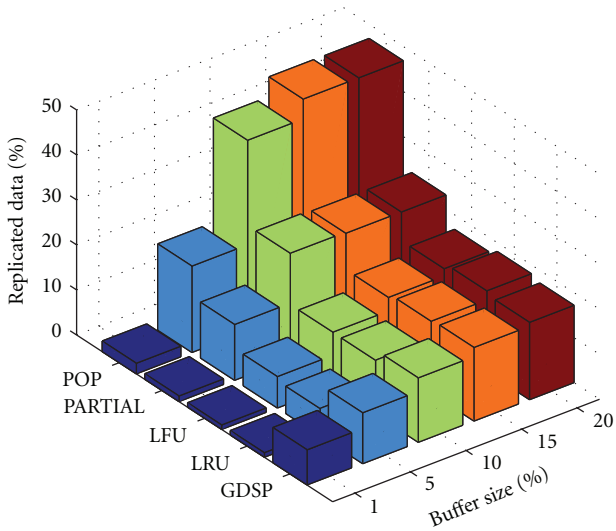
where $\mathrm{HR}_{\mathrm{NorepWith}}$ is the collected hit rate in a system with churn but without replicated data, and $\mathrm{HR}_{\mathrm{Without}}$ is the hit rate in a system without churn.

Figure 10 shows the decreasing ratio for all studies' policies. We have noted that in spite of the enlargement of the system storage capacities, the decreasing ratio is around 40% when the exponential churn model drives the joining and leaving dynamics. For peers under the heavy tail churn model, this decreasing ratio is around 20% (see Figure 10(b)).

Compared to results showed in Figure 7, the performance of policies under both churn models reduces by 50%.

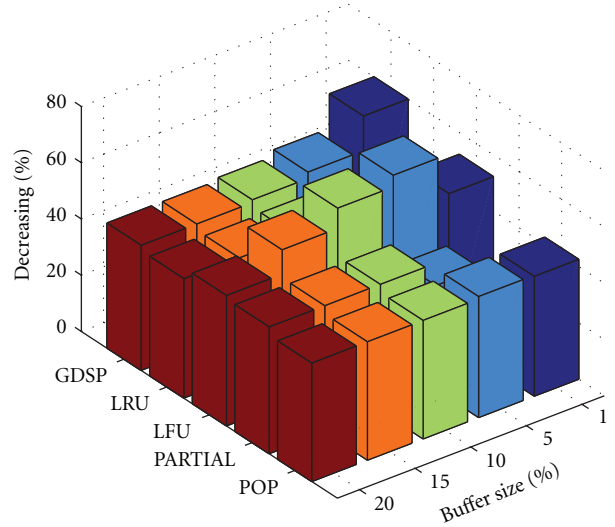



(a) Exponential churn model

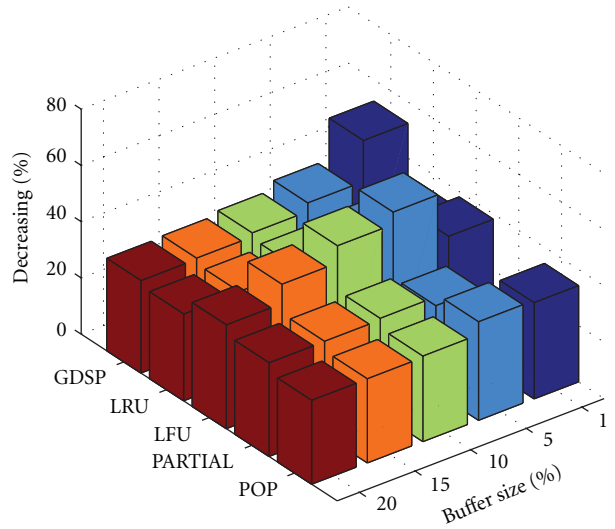



(b) Heavy tail churn model

Figure 9: The amount of replicated data handled by policies.




(a) Exponential churn model




(b) Heavy tail churn model

Figure 10: The decreasing on hit rate in a nonreplica system (see (7)).

This suggests that (i) an enlarged storage capacity has limited impact over policies performance and (ii) a controlled amount of replicated data tends to improve the performance of policies. In fact, it is under evaluation an algorithm to support replicating. Peers that just initiated the leaving procedure will use this algorithm to spread objects' metadata in order to support the decisions made by other peers during the evicting process.

## 8. Related Work

Several approaches (e.g., [8]) have been proposed in the literature so as to deal with video caching. In these approaches, whether a video or some part of it cannot be found in local proxies, it can be requested from the original central provider. In a P2P video system, however, there is no such central video server nor does any peer guarantee the availability of any data it caches.

Zink et al. [9] and Cha et al. [3] strongly suggest that metrics such as object popularity has to be considered in-cache object management policies. These studies show that popular and nonpopular videos access ratio is 2 in 3, despite popular video collection is being made of just 3.7% of the whole video collection. Although Zink et al. [9] show that local popularity has a weak correlation with the global popularity, still most of the accessed videos are the local popular ones.

Kulkarni and Devetsikiotis [10] propose the design of a distributed cache similar in principle to a content distribution network (CDN). The goal is to select some of the most valuable objects and bring them closer to the clients requesting them so that redundant load on the network is reduced. Social network analysis techniques were used to

identify the most valuable objects that should be cached. While Kulkarni's work focus is to determine the measurements that can be used to define the objects popularity, our goal is to verify the effectiveness of such measurements in the implementation of a cache approach based on P2P systems.

The behavior of P2P system under churn is one of the most fundamental issues of P2P research (e.g., [6, 11, 12]). Several approaches (e.g., [11, 13]) have dealt with churn by investigating its characteristics (e.g., median session length) in large-scale P2P systems. Gummadi et al. [11] measure session lengths by monitoring a router at the University of Washington. Sripanidkulchai et al. [13] study the live streaming workload of a large-content delivery system and present an analysis characterizing popularity, arrival process, and session length.

## 9. Conclusions

In this paper we studied object management policies in a peer-to-peer network context. While traditional object management policies were built around only three object proprieties, that is, aging, size, and frequency, we built our policy around user-generated content and metadata made available by content providers. We used video popularity and took advantage of video-encoding techniques to build our policy. We have carried out a set of simulation experiments so as to evaluate the performance of our approach. In the first part of our simulation experiments, we have observed that our approach outperforms the traditional one and consequently will reduce the capacity demand poses over the community output link by increasing the hit rate of community demands and optimizing the network resources available.

We have studied as well the impact of the churn on the object management policies. We evaluated the enlargement of peer's storage capacity as an action to keep policies performance since the system storage capacity will be affected by the churn. Though the gap among a policy's performance shrinks as we enlarged the storage capacity, it does not impact proportionally a policy performance.

Also, we have carried out studies to look into how replicated data impact the performance of object management policies. We found that the worst case scenario, in terms of content availability, is for a system without replicated data. Despite the enlargement of the storage capacity, in general, we have not seen improvements on policies performance for such systems. On the other hand, replicating due to the churn improved the performance of policies to a certain level. However, we have shown from the POP policies performance, whether the amount of duplicated data is under policy control, the content availability could be greatly improved.

As future works, we have looked into mechanisms to control the amount of replicated data through the system and investigated whether or not incentive-based peer participating mechanisms, developed for other content distribution systems, could be applied in such hybrid P2P-CDN system.

## References

[1] S. Jin and A. Bestavros, "Popularity-aware greedy dual-size web proxy caching algorithms," in *Proceedings of the The 20th International Conference on Distributed Computing Systems (ICDCS '00)*, IEEE Computer Society, Washington, DC, USA, 2000.

[2] M. Hefeeda and O. Saleh, "Traffic modeling and proportional partial caching for peer-to-peer systems," in *Proceedings of the IEEE/ACM Transactions on Networking*, vol. 16, pp. 1447–1460, IEEE Press, Piscataway, NJ, USA, 2008.

[3] M. Cha, H. Kwak, P. Rodriguez, Y. Y. Ahnt, and S. Moon, "I tube, you tube, everybody tubes: analyzing the world's largest user generated content video system," in *Proceedings of the 7th ACM SIGCOMM Internet Measurement Conference (IMC '07)*, pp. 1–14, October 2007.

[4] A. Crespo and H. Garcia-Molina, "Semantic overlay networks for P2P systems," in *Agents and Peer-to-Peer Computing*, vol. 3601 of *Lecture Notes in Computer Science*, pp. 1–13, Springer, Berlin, Germany, 2005.

[5] F. Wang, J. Liu, and Y. Xiong, "Stable peers: existence, importance, and application in peer-to-peer live video streaming," in *Proceedings of the 27th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '08)*, pp. 1364–1372, IEEE Computer Society, Phoenix, Ariz, USA, 2008.

[6] Z. Yao, D. Leonard, X. Wang, and D. Loguinov, "Modeling heterogeneous user churn and local resilience of unstructured P2P networks," in *Proceedings of the IEEE International Conference on Network Protocols, (ICNP '06)*, pp. 32–41, Washington, DC, USA, 2006.

[7] A. Bezerra, C. Melo, D. Vieira, Y. Ghamri-Doudane, and N. Fonseca, "A content-oriented web cache policy," in *Proceedings of the IEEE Latin-American Conference on Communications, (LATINCOM '09)*, pp. 1–6, September 2009.

[8] Z. Miao and A. Ortega, "Scalable proxy caching of video under storage constraints," *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 7, pp. 1315–1327, 2002.

[9] M. Zink, K. Suh, Y. Gu, and J. Kurose, "Watch global, cache local: youtube network traffic at a campus network—measurements and implications," in *Proceedings of the Multimedia Computing and Networking Conference, (MMCN '08)*, January 2008.

[10] V. Kulkarni and M. Devetsikiotis, "Communication timescales, structure and popularity: using social network metrics for youtube-like multimedia content distribution," in *Proceedings of the IEEE International Conference on Communications, (ICC '10)*, May 2010.

[11] K. P. Gummadi, R. J. Dunn, S. Saroiu, S. D. Gribble, H. M. Levy, and J. Zahorjan, "Measurement, modeling, and analysis of a Peer-to-peer file-sharing workload," in *Proceedings of the 19th ACM Symposium on Operating Systems Principles, (SOSP '03)*, pp. 314–329, ACM, New York, NY, USA, October 2003.

[12] D. Leonard, Z. Yao, V. Rai, and D. Loguinov, "On lifetime-based node failure and stochastic resilience of decentralized peer-to-peer networks," *IEEE/ACM Transactions on Networking*, vol. 15, no. 3, pp. 644–656, 2007.

[13] K. Sripanidkulchai, B. Maggs, and H. Zhang, "An analysis of live streaming workloads on the internet," in *Proceedings of the ACM SIGCOMM Internet Measurement Conference, (IMC '04)*, pp. 41–54, New York, NY, USA, October 2004.

Hindawi Publishing Corporation
International Journal of Digital Multimedia Broadcasting
Volume 2012, Article ID 716780, 12 pages
doi:10.1155/2012/716780

*Research Article*

# Multi-Objective Genetic Algorithm for Task Assignment on Heterogeneous Nodes

**Carolina Blanch Perez del Notario, Rogier Baert, and Maja D'Hondt**

*SSET Department of IMEC, Kapeldreef 75, 3001 Leuven, Belgium*

Correspondence should be addressed to Carolina Blanch Perez del Notario, blanch@imec.be

Received 30 April 2011; Revised 5 September 2011; Accepted 20 September 2011

Academic Editor: Yifeng He



Task assignment in grid computing, where both processing and bandwidth constraints at multiple heterogeneous devices need to be considered, is a challenging problem. Moreover, targeting the optimization of multiple objectives makes it even more challenging. This paper presents a task assignment strategy based on genetic algorithms in which multiple and conflicting objectives are simultaneously optimized. Specifically, we maximize task execution quality while minimizing energy and bandwidth consumption. Moreover, in our video processing scenario; we consider transcoding to lower spatial/temporal resolutions to tradeoff between video quality; processing, and bandwidth demands. The task execution quality is then determined by the number of successfully processed streams and the spatial-temporal resolution at which they are processed. The results show that the proposed algorithm offers a range of Pareto optimal solutions that outperforms all other reference strategies.

## 1. Introduction

Nowadays multimedia applications such as multicamera surveillance or multipoint videoconferencing, are increasingly demanding both in processing power, and bandwidth requirements. In addition, there is a tendency towards thin client applications where the processing capacities of the client device are reduced and the tasks are migrated to more powerful devices in the network.

In this respect, grid computing can integrate and make use of these heterogeneous computing resources which are connected through networks, overcoming the limited processing capabilities at a client's device.

In the context of distributed media processing we can think of scenarios such as video control rooms where multiple video streams are processed and simultaneously displayed. One way to downscale the processing and bandwidth requirements at the displaying device is by transcoding the video streams at the servers to lower temporal or spatial resolutions. This is done, however, at the cost of a degraded perceived video quality and an increased processing cost at the server. Therefore, in grid computing we may need to optimize and trade off multiple objectives, targeting for instance quality maximization of the stream execution and minimization of the energy consumption on the client/servers simultaneously. In this respect, implementing a suitable strategy for task assignment/scheduling becomes crucial for achieving a good performance in grid computing. This subject has been thoroughly studied in literature, and various heuristic approaches have been widely used for scheduling. In particular, Genetic Algorithms (GAs) have received much attention as robust stochastic search algorithms for various optimization problems. In this context, works such as [1–3] have used Genetic Algorithms to approach task scheduling within a device. In [1, 2], genetic algorithms are combined with heuristics such as "Earliest Deadline First" to assign tasks onto multiple processors. The work in [3] extends the analysis to heterogeneous processors where the algorithm can also determine the optimal number of processors, given a number of tasks. In [4, 5], the work is extended to multiple processing nodes in the network. This way, in [4], a thorough study is done on the performance of different GA operators on a scheduling problem on grid computing systems. Heterogeneous processors are considered with the target to minimize the makespan and flow time of the tasks.

The authors in [5] use also GA to address a similar objective in grid computing. As in [4] neither data transmission nor resource cost is considered.

In [6], a combination between GA and ACO algorithms is presented for task scheduling among multiple nodes but no bandwidth considerations are made. In [7], the authors use evolutionary fuzzy systems to solve job scheduling in decentralized computational grids where jobs between grid clusters are exchanged. The average response time per job is minimized but again the overhead of data transfers is not considered. The work in [8] also uses a genetic-based algorithm to assign multiple tasks in grid computing and minimize the make-span in the task execution but unlike previous works, the transmission time to the processing node is considered. Similar considerations are taken in [9] where the authors propose a strategy based on the Ant Colony Optimization algorithm (ACO) to effectively assign tasks to computing resources, given the current load conditions of each computing resource and network status.

Note that the presented works only consider single objective optimization. It is in works such as [10–16] where GAs are used to address multiple objectives in the resolution of scheduling problems. This way, in [10], the authors address the Job Shop scheduling problem while targeting multiple objectives, namely, minimal make-span and minimal machine workload. However, the case study addressed is very simple and only homogeneous processors are considered. The work in [11] addresses the flow-shop scheduling problem with an adaptive genetic algorithm. Pareto fronts are used to guide a multiple-objective search: the total completion time and total tardiness. As in our work, multiple objectives are addressed, however, task assignments at system level and bandwidth limitations are not considered. The authors in [12] consider heterogeneous processors and a multiobjective GA targets in this case the minimization of the make-span, flow-time, and reliability. The work in [13] is an early work on applying GA for multiobjective optimization, in this case to minimize both task execution and power during cosynthesis of distributed embedded systems. In [14], the performance of a NSGA-II-based evolutionary algorithm is compared to the true Pareto front of solutions found with the Branch and Bound method. The objectives targeted are the make-span and the task completion time. Finally, in [15] the method of particle swarm optimization is combined with evolutionary method outperforming typical genetic algorithms.

However, in none of these works [12–15], there is any consideration made over bandwidth constraints. Only in [16] the network resources in terms of bandwidth and latency are considered. In this case, an Ant Colony Optimization algorithm is developed to address grid scheduling.

In our approach, we use GAs to target multiple objectives for task assignment in grid computing and we consider bandwidth availability between nodes. Moreover, in comparison with all related work presented, in our analysis, we introduce an extra dimension on the task assignment problem by considering the downscaling of the video streams to lower spatial/temporal resolution. This offers a tradeoff between bandwidth and processing constraints on one hand and perceived video quality on the other hand. By doing this the effective system capacity to process tasks is increased while a graceful degradation of the video stream quality is allowed. Additionally, we target multiple objectives such as task quality maximization, client's energy minimization, and minimization of the bandwidth usage.

The rest of the paper is structured as follows. Section 2 describes the scenarios for distributed media processing considered. Section 3 describes the basic strategies for task assignment as well as the strategy for quality maximization while Section 4 introduces the strategy based on genetic algorithms. We present the results and compare the performance of the different strategies in Section 5. Finally, Section 6 concludes this work.

## 2. Scenario of Distributed Media Processing

In the context of distributed video processing, we are considering a scenario such as the one of a video control room. Several video contents are streamed towards the client device, where the content is visualized, while the required video processing can be distributed between the client and other processing nodes such as servers.

We assume all processing nodes to be heterogeneous with a different amount of processing resources, such as CPUs and GPUs. In addition, we assume that the client's device has more limited processing capacities than the server nodes. Concretely, we consider 4CPUs and 1GPU at each server node, while only 2CPUs at the client node. We assume moreover that multiple codec implementations for these different processor types are available. To overcome the limited processing at the client node, we perform distributed processing over other nodes in the network. In this case, the decoding task is executed at a server, and the resulting output (raw video) is transmitted to the client's device. Note that this highly increases the bandwidth requirements, which should fit in the maximum available bandwidth towards the client that we assume of 1 Gbps and shared from any server node to the client node. Therefore, to fit both processing and bandwidth requirements, one possibility is to trans-code (decode and reencode at a lower temporal or spatial resolution) the video streams at the server's side. This lowers both its bandwidth and decoding processing requirements at the end device at the cost of a reduced perceived quality and increased server processing.

The following section describes the task assignment strategies used in the scenario described.

## 3. Single Objective Assignment Strategies

An efficient task assignment strategy is a key element in the context of distributed grid computing. In this section, we describe the assignment strategies that we implement for comparison with our evolutionary-based approach.

*Round-Robin Strategy (RR).* The stream processing tasks are assigned in turns on the different available processing elements, that is, client device and server nodes.

*Max-Min Heuristic (MM).* This is a well-known heuristic [17] that assigns the most demanding stream processing tasks first on the processor that is going to finish them the earliest.

*TransCode-All Strategy (TA).* We assign all video streams to be spatially trans-coded at the server nodes. This lowers the processing requirements for decoding at the client devices while it also reduces the bandwidth usage. However, this happens at the cost of a reduced quality of the video streams. As trans-coding is an intensive task (decoding plus encoding) the trans-coding of the streams is evenly distributed among the available servers (by means of round robin) to avoid processing overload of a server.

*Strategy Maximum Quality (MaxQ).* In addition to the presented strategies, we implement a strategy that targets the maximization of the quality of the stream assignment. We describe this strategy next for the case of 1 server and 1 client node, where $P_{\text{client}}$ and $P_{\text{server}}$ are the total processing cost at client and server for the current task assignment, and $P_{\text{client}}^{\max}$ and $P_{\text{server}}^{\max}$ are the maximum processing capacity at the client node and server respectively. In a similar way, BW is the bandwidth required by the current task assignment while $\text{BW}^{\max}$ corresponds to the maximum available bandwidth.

We consider that the assignment and execution, $t_i$, of video stream "$i$" can take one of the following values:

- $c$: stream assigned to be decoded at the client at original temporal and spatial resolution.
- $s$: stream assigned to be decoded at the server at original temporal and spatial resolution.
- temp: stream transcoded to lower temporal resolution at the server.
- spat: stream transcoded to lower spatial resolution at the server.
- comb: stream transcoded to both lower temporal and spatial resolution at the server.

This way, our task assignment consists of a set of $t_i \in \{c, s, \text{temp}, \text{spat}, \text{comb}\}$ for every $i \in \{1, \ldots, N\}$ where $N$ is the total number of video streams to be processed.

We want to find a task assignment solution whose bandwidth and processing demands at client and server fit within the bandwidth and processing constraints:

$$\begin{aligned} \text{BW} &= \sum_i \text{BW}_i \leq \text{BW}^{\max}, \\ P_{\text{client}} &= \sum_i P_{i,\text{client}} \leq P_{\text{client}}^{\max}, \\ P_{\text{server}} &= \sum_i P_{i,\text{server}} \leq P_{\text{server}}^{\max}, \\ \forall i \mid t_i &\in \{c, s, \text{temp}, \text{spat}, \text{comb}\}. \end{aligned} \tag{1}$$

Algorithm 1 is described in the following paragraphs and table.

In Step 1, the algorithm assigns as many stream decoding tasks as possible to execute on the client device; this number of tasks is constrained by the processing power at the device.

*Step 1*: *Maximize* $\{k \mid t_i = c\ \nabla i = \{1, \ldots, k\}, P_{\text{client}} \leq P_{\text{client}}^{\max}\}$
*Step 2*: *Set* $t_i = s\ \nabla i = \{(k+1) \cdots N\}$
*WHILE* ($P_{\text{client}} \geq P_{\text{client}}^{\max}$ or $P_{\text{server}} \geq P_{\text{server}}^{\max}$ or $\text{BW} \geq \text{BW}^{\max}$)
  *Step 3: IF* $\exists i = \max_i\{\text{BW}_i \mid t_i = s\}$
    *THEN* $t_i = temp$
  *ELSE IF* $\exists i = \max_i\{\text{BW}_i \mid t_i = temp\}$
    *THEN* $t_i = spat$
  *ELSE IF* $\exists i = \max_i\{\text{BW}_i \mid t_i = spat\}$
    *THEN* $t_i = comb$
  *END*
  *Repeat* Step 3 *until* $\text{BW} \leq \text{BW}^{\max}$
  *Step 4: IF* ($P_{\text{server}} \geq P_{\text{server}}^{\max}$ *and* $t_i = comb\ \nabla i = \{1, \ldots, N\}$
    *THEN* ***STOP***
  *ELSEIF* $P_{\text{client}} \geq P_{\text{client}}^{\max}$ *THEN*
    *IF* $\exists i \mid t_i = c$ *THEN*
      *Set* $t_i = s$
    *ELSE* ***STOP***
  *END*
*END*

Algorithm 1

In Step 2, the remaining tasks, exceeding the processing power at the client device, are assigned for processing at the server.

Then, we check if the current assignment meets bandwidth and processing constraints. While either bandwidth or processing constraints at client or server are not met, the algorithm will gradually transcode video tasks to lower temporal or spatial resolution at the server (done in Step 3) or will migrate some of the decoding tasks from the client device to the server (done in Step 4). This process continues till the assignment fits the system bandwidth and processing constraints.

In Step 3 we proceed as follows.

(i) Find those stream which are currently assigned at original temporal and spatial resolution to the server ($t_i = s$). From those, pick the stream with the biggest BW demand and transcode it to lower temporal resolution ($t_i = \text{temp}$).

(ii) If there are no streams available at full temporal resolution, then we take a stream at lowered temporal resolution ($t_i = \text{temp}$) with the highest bandwidth demand and transcode it to a lower spatial resolution ($t_i = \text{spat}$).

(iii) If all streams have been spatially transcoded, then we pick one of them (at highest bandwidth) and transcode it both temporally and spatially ($t_i = \text{comb}$).

Note that at this point (Step 3), we are trying to find those stream tasks ($t_i$) that demand the maximum bandwidth (generally also the highest processing) in order to reduce the bandwidth demands by trans-coding the minimum amount of streams.

This procedure is repeated till the bandwidth constraint is met.

Finally, in Step 4, if the processing constraints at the client are exceeded we migrate one client task to the server side. If at the server's side the processing constraints are not met and all streams have been spatially and temporally transcoded, the assignment loop is stopped. It is not possible to downscale the stream tasks further, and therefore we cannot find an assignment that satisfies all constraints while processing all streams.

*3.1. Evaluation of Stream Assignment Cost.* If the task assignment exceeds any of the system constraints in (1), then the execution of some tasks will fail. This way, an individual stream processing task fails when it does not fit within the available processing or bandwidth resources. For instance, the node where the stream is assigned may not have sufficient processing resources, or even if the processing is completed at the server, the available bandwidth could be insufficient to deliver the server's output to the client causing the stream processing to fail.

Related to this, we can attach to each assignment solution a corresponding cost in terms of end video quality, bandwidth usage, and energy consumption. This cost is determined by how many stream tasks are successfully completed and how (on which device and at what spatial-temporal resolution) they are executed.

Therefore, for a specific stream assignment solution we first need to estimate which stream processing tasks can be successfully completed and which will fail due to not meeting current processing and bandwidth constraints. Then, depending on the specific execution of each individual stream processing, we can attach a cost, in terms of quality, consumed bandwidth, and energy at the client's side, as defined in Table 1.

Note that the data in Table 1 corresponds to streams of Standard Definition (SD, 720 × 480 pixels). We also consider streams of Common Intermediate Format (CIF, 352 × 288 pixels) and High Definition (HD, 1920 × 1080 pixels) where bandwidth and energy costs are scaled accordingly with respect to SD resolution.

A stream processing fails when it does not fit within the available processing, or bandwidth resources. For instance, the node where the stream is assigned may not have sufficient processing resources or even if the processing is completed at the server, the available bandwidth could be insufficient to deliver the server's output to the client causing the stream processing to fail.

We attach successfully processed streams a quality value of 1 when the content is displayed at the client at its original temporal and spatial resolution. If the video stream is downscaled to a lower temporal/spatial resolution in order to fit bandwidth or processing constraints, the perceived video quality will be slightly degraded, and therefore, we attach a lower quality value. This favors that to maximize the streams quality, assignment solutions where the original spatial and temporal resolution of the streams are kept are preferred. Note that the quality value of any stream at its original resolution (CIF, SD, or HD) is identical; only in transcoding, we consider the quality degraded; that is, an HD-streamed spatially transcoded (to SD) is attached a 0.8 quality (distortion of 0.2) while a stream at original SD resolution is attached the maximum quality of 1 (0 distortion).

Table 1: Definition of task execution costs (SD resolution).

| Stream Execution | Quality $TQ_i$ | Bandwidth $TBW_i$ | Energy $TE_i$ |
|---|---|---|---|
| Failed execution | 0 | 0 Mbps | 0 |
| Decoding at client | 1 | 20 Mbps | 1 |
| Decoding at server | 1 | 146 Mbps | 0 |
| Temporal transcoding | 0.9 | 10 Mbps | 0.5 |
| Spatial transcoding | 0.8 | 7 Mbps | 0.25 |
| Temporal and Spatial transcoding | 0.7 | 3.5 Mbps | 0.12 |

The bandwidth cost per stream is also dependent on how the stream processing is performed. This way, if decoding is performed at the server's side, the stream is transmitted raw to the client, which highly increases the bandwidth requirements. On the contrary, if the stream is transcoded at a server to a lower spatial or temporal resolution, the bandwidth requirements are reduced. For the sake of simplicity, we assume the same bandwidth cost for all video streams with the same spatial-temporal resolution. In addition, we consider that reducing the temporal resolution from 30 frames per second to 15 approximately reduces the bandwidth by half. Similarly, we assume that reducing the spatial resolution to the immediate lower resolution roughly reduces the bandwidth to approximately one-third of the original resolution.

Finally, in terms of energy/processing cost at the client's device, we assume that the energy cost is negligible when the video decoding task is executed on a server, and the raw output video stream is merely transmitted to the client device for display. When the decoding task is executed at the client, the corresponding energy cost is dependent on the temporal and spatial resolution of the decoded stream. We assume that decoding a video sequence at 15 fps requires approximately half of the processing/energy than decoding the same sequence at 30 fps. In a similar way, when the spatial resolution is lowered, for example from SD to CIF, we can roughly assume 1/4 of the decoding energy costs. Finally, the combination of lowering temporal and spatial resolution corresponds to a decoding cost of 18 of the original resolution. In case of a failed task execution, we assume no processing effort at the client's side and therefore no energy consumption. Note that the values in Table 1 are taken as approximate and reference values and have no impact on the relative performance of the assignment strategies.

To obtain the total quality TQ, bandwidth TBW, or energy cost TE for the complete assignment, we simply need to sum all individual stream processing costs as follows:

$$\text{Quality TQ} = \sum_{i=1}^{N} \text{TQ}_i \tag{2}$$

or equivalently

$$\text{Distortion TDist} = \sum_{i=1}^{N} \text{Max Quality} - \sum_{i=1}^{N} \text{TQ}_i, \tag{3}$$

$$\text{Bandwidth TBW} = \sum_{i=1}^{N} \text{TBW}_i, \tag{4}$$

$$\text{Energy TE} = \sum_{i=1}^{N} \text{TE}_i, \tag{5}$$

where $N$ is the total number of streams considered in the assignment.

## 4. Multiobjective Genetic Algorithm

The heuristic and strategies presented in the previous section target at most one single objective optimization. However in practice we may want to optimize multiple objectives simultaneously. For instance, we may need to maximize the video streams quality while minimizing the bandwidth usage and the energy cost at the client. This multiobjective optimization is challenging, especially when multiple heterogeneous nodes and multiple ways of processing the streams (decoding, trans-coding) are considered. To achieve this, we base ourselves on genetic algorithms and use the concept of Pareto fronts of solutions. This allows us to obtain a set of Pareto optimal assignment solutions from which we can choose the solution that best meets the constraints or our preferences towards a certain objective. In addition, a genetic algorithm is a flexible tool where the target objective can be easily modified. The remainder of this section describes how the genetic algorithm is implemented.

*4.1. Genetic Algorithm Structure.* A genetic algorithm (GA) is a search heuristic that mimics the process of natural evolution. In a genetic algorithm, a population of strings (called chromosomes), which encode candidate solutions (called individuals of the population) to an optimization problem, evolves toward better solutions. The evolution usually starts from a population of randomly generated individuals and happens in generations. In each generation, the fitness of every individual in the population is evaluated, multiple individuals are selected from the current population (based on their fitness), and modified (recombined and possibly randomly mutated) to form a new population. The new population is then used in the next iteration of the algorithm. Generally, each generation of solutions improves the quality of its individuals. The algorithm terminates when either a maximum number of generations has been produced, or a satisfactory fitness level has been reached. The structure of our genetic algorithm can be summarized as follows.

*Step 1.* Initialize the population of chromosomes.

*Step 2.* Evaluate each chromosome with the *fitness* function.

*Step 3. Crossover* operation: select parents according to fitness and create new children chromosomes.

*Step 4.* Random *mutation* of chromosomes.

*Step 5.* Elitist *selection:* retain fittest chromosomes among those of the old generation and the new ones resulting from Steps 3 and 4.

*Step 6.* Repeat Steps 2 to 5 till termination condition is reached.

*4.2. Representation of the Solution Domain.* Traditionally, solutions are represented in binary as strings of 0s and 1s, but other encodings are also possible. In our case, we use a decimal representation. Each possible stream assignment solution is represented as a chromosome, which is composed of several gens. In our case, the length of the chromosome is equal to the number of streams that need to be scheduled in the system. Each of the genes in the chromosome represents the node that is going to process the stream and how it is going to be processed. Table 2 gives a description of the meaning of each possible gene value in the chromosome. In this example, the available processing nodes are the client device S0 and two server nodes S1 and S2.

Figure 1 shows an example of chromosome (assignment solution). By looking at the gene description in Table 2, we can see that the corresponding stream task assignment would be as follows: the first two streams ("2") are decoded at server S1, the third and fourth streams ("1") are executed at the client device S0, the next three streams are transcoded at server S2 to a lower spatial resolution ("8"), and the last stream is processed at server S1 where it is transcoded to lower temporal resolution ("3").

We now detail how to compute the cost of the assignment solution in Figure 1. Assuming all streams are of SD resolution, the corresponding cost of this assignment according to Table 1 would be the following.

For streams 1 and 2 decoded at the server at full resolution and transmitted raw to the client:

$$\begin{aligned} \text{BW}_1 = \text{BW}_2 &= 146\,\text{Mbps}, \\ \text{TQ}_1 = \text{TQ}_2 &= 1, \\ \text{TE}_1 = \text{TE}_2 &= 0. \end{aligned} \tag{6}$$

For streams 3 and 4 decoded at the client device at full resolution:

$$\begin{aligned} \text{BW}_3 = \text{BW}_4 &= 20\,\text{Mbps}, \\ \text{TQ}_3 = \text{TQ}_4 &= 1, \\ \text{TE}_3 = \text{TE}_4 &= 1. \end{aligned} \tag{7}$$

For streams 5, 6, and 7 spatially transcoded at a server:

$$\begin{aligned} \text{BW}_5 = \text{BW}_6 = \text{BW}_7 &= 7\,\text{Mbps}, \\ \text{TQ}_5 = \text{TQ}_6 = \text{TQ}_7 &= 0.8, \\ \text{TE}_5 = \text{TE}_6 = \text{TE}_7 &= 0.25 \end{aligned} \tag{8}$$

For stream 8 that is temporally transcoded at a server before being sent to the client:

$$\text{BW}_8 = 10\,\text{Mbps}, \qquad \text{TQ}_8 = 0.9, \qquad \text{TE}_8 = 0.5. \tag{9}$$

Therefore, the total value of the assignment in the three-dimension space is

$$\sum_i \text{BW}_i = 363\,\text{Mbps}, \qquad \sum_i \text{TQ}_i = 7.3, \qquad \sum_i \text{TE}_i = 3.25. \tag{10}$$

Table 2: Description of genes.

| Gene value | Meaning on task execution |
|---|---|
| "1" | Decoded at client device S0 |
| "2" | Decoded at S1 and transmitted to S0 |
| "3" | Transcoded at S1 to lower temporal resolution |
| "4" | Transcoded at S1 to lower spatial resolution |
| "5" | Transcoded at S1 to lower spatial-temporal resolution |
| "6" | Decoded at S2 and transmitted to S0 |
| "7" | Transcoded at S2 to lower temporal resolution |
| "8" | Transcoded at S2 to lower spatial resolution |
| "9" | Transcoded at S2 to lower spatial-temporal resolution |

| 2 | 2 | 1 | 1 | 8 | 8 | 8 | 3 |
|---|---|---|---|---|---|---|---|

Figure 1: Example of chromosome.

*4.3. Initialization of GA Population.* In general terms we initialize the population of assignment solutions by random generation. We also include in the initial population solutions that contribute to distribute the tasks processing evenly among the existing processing elements as well as solutions that imply transcoding of all processing tasks (as this may facilitate convergence to suitable assignments in high load scenarios). By doing so, we are making sure that certain potentially useful features are present in the population.

With respect to the population size, its optimal value is highly dependent on the scenario dimensions. In our case, we experimented with populations of size 10 to 40 and determined experimentally that a population of size 30 was suitable for the considered scenarios.

In addition, in a dynamic scenario where the number of tasks to be processed may be varying over time, we can improve convergence by reusing previously found solutions as part of the initial population for a new scenario. For instance, if the stream tasks to be processed increase from $N$ to $N+1$, then we can use the assignment solutions found for $N$ streams as initial population for the assignment of $N+1$ streams, while a random assignment is added for the extra $N+1$th stream. This helps the algorithm converge and finds an optimal solution. Moreover, it helps preserve the previous assignment solution as it minimizes the change of tasks assignments in the system.

*4.4. Fitness Function.* The goal of the fitness function is to evaluate how good an assignment solution is with respect to the defined target objectives. If we consider the optimization of a single objective, for instance, maximization of the video quality, we can define the fitness function as the quality value of every assignment:

$$\text{Fitness} = \text{Quality} = \sum_{i=1}^{N} TQ_i \qquad (11)$$

$N$ = total number of streams.

This way, the fittest assignments are those with the highest video stream quality.

If we are considering multiple objectives such as video quality, bandwidth usage, and energy consumption at the client device, a particular assignment solution will result in a certain value in these three axes. In other words, each assignment solution can be represented as a point in the multiobjective space with the objective values (TDist, TBW, and TE).

Each of these values is obtained as the sum of distortion, bandwidth and energy for all $N$ stream tasks considered:

$$\text{TDist} = \sum_{i=1}^{N} \text{TDist}_i, \quad \text{TBW} = \sum_{i=1}^{N} \text{TBW}_i, \quad \text{TE} = \sum_{i=1}^{N} \text{TE}_i. \qquad (12)$$

With the task distortion related to the task quality as

$$\text{TDist}_i = \text{Max}\, Q_i - \text{TQ}_i. \qquad (13)$$

To minimize multiple objectives, we would like then to minimize the following function where $w1$, $w2$, and $w3$ are the weights for the different objectives:

$$f(\text{dist}, \text{BW}, E) = w1 * \text{TDist} + w2 * \text{TBW} + w3 * \text{TE}. \qquad (14)$$

Minimizing (14) is then equivalent to maximizing the following fitness function in our GA:

$$\text{Fitness} = \frac{1}{f(\text{dist}, \text{BW}, E)}. \qquad (15)$$

To avoid the need of weighting factors in the fitness function, we use the concept of Pareto points for the multiobjective optimization. Pareto points are optimal tradeoffs in the multiobjective space and obtaining the set of Pareto points from all assignment solutions is equivalent to exploring a range of weights in (14)-(15).

We are therefore interested in obtaining a range of Pareto optimal solutions in said multiobjective space. In addition, we evaluate the fitness of an assignment solution according to how close the solution is to a Pareto point or to the actual Pareto envelope.

In Figure 2, we can see that this favors that non-Pareto solutions like A, which lie close to the Pareto front, are selected to breed and mutate hopefully generating Pareto points within new areas of the Pareto front. This helps newer generations of Pareto front spread over the bidimensional space without losing diversity and concentrating around the specific area of the initial Pareto points.

To evaluate the fitness of each solution point, we compute the Euclidean distance from each point to the closest point in the hypothetical Pareto front. In a three-dimensional objective space, this is expressed as

$$\text{Fitness}(i) = \frac{1}{\min}(\text{Euclid}(i,j)) = \frac{1}{\min}\left(\sqrt{(i_1 - j_1)^2 + (i_2 - j_2)^2 + (i_3 - j_3)^2}\right) \quad \nabla j \in \text{Pareto front}, \qquad (16)$$

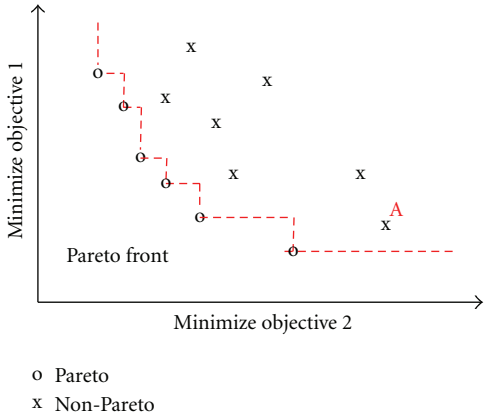



Figure 2: Pareto front of solutions.

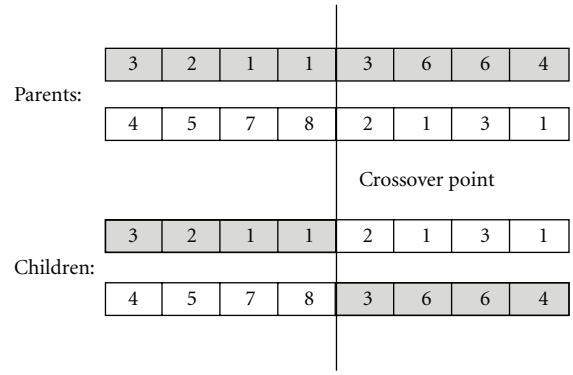



Figure 3: Single-point crossover.

where every point has the following representation in the three-dimensional multiobjective space $i = \langle i_1, i_2, i_3 \rangle$. The closer a point is to this Pareto front, the fitter it is considered. Note that Pareto points, already belonging to the Pareto envelope, have the minimum distance to it and therefore are assigned the highest fitness values. This guarantees that they are always kept for the next generation of the GA.

*4.5. Selection of Individuals According to Fitness.* During each successive generation, a proportion of the existing population of solutions is selected to breed a new generation. Individual solutions can be selected through a fitness-based process, where fitter solutions (as measured by the fitness function) are typically more likely to be selected. In our case, we do not use a probabilistic selection but an elitist selection, that is, the fittest members of the population are used to breed a new population. Moreover, after crossover and generation of new child solutions, we apply again elitist selection and retain in the solution space those solutions that are the fittest among the parents and the newly generated children solutions. We use the fitness function as described in the earlier section. In practice, this means that for the selection of the parent chromosomes during the crossover and mutation steps, we select the Pareto optimal points from the pool of chromosome/solutions (as these are the fittest points in our space) and from the non-Pareto points, we take the fittest ones.

The elitism we apply in the selection is similar to the one applied in the Nondominated sorting GA or NSGA [18] in the sense that the Pareto or nondominated solution points are identified and given the highest fitness value. Our approach differs however in the treatment of the non-Pareto solutions, in our case, the fitness of these points is computed based on the distance to the Pareto front, more similar to the Strength Pareto Evolutionary Algorithm (SPEA). With respect to SPEA [19], we apply a similar elitist selection, after both crossover and mutation as the Pareto solutions are always kept for the next generation. However, unlike SPEA, we do not maintain a fixed size for the Pareto population, and to avoid a too early convergence of the algorithm, we allow a high degree of mutation among the solutions in the Pareto front.

*4.6. Evolution: Crossover and Mutation.* The *crossover* step consists in creating a second generation of solutions from an initial population of solutions. For each new solution to be produced, a pair of "parent" solutions is selected for breeding from the current population. By producing a "child" solution using the crossover, a new solution is created which typically shares many of the characteristics of its "parents". In our algorithm, crossover is implemented as single-point crossover. This is, a random point is selected in the parent chromosome, and the information before and after that point is exchanged and recombined between the two parents creating in this way two child solutions that share the parent's characteristics. The single-point crossover is represented in Figure 3.

Crossover is applied on a percentage of the population given by the *crossover rate*. Out of the new population of "parents" and "child" chromosomes, we apply an elitist selection and keep a population with size of the initial population with the fittest individuals. Generally, the average fitness of the new population will have increased, since only the best individuals from the first generation are selected for breeding. After crossover, the *mutation* step is implemented. The purpose of mutation in GAs is preserving and introducing diversity. Mutation should allow the algorithm to avoid local minima by preventing the population of chromosomes from becoming too similar to each other, thus slowing or even stopping evolution. Mutation is implemented by randomly changing some of the genes in the chromosome with a probability given by the *mutation rate*.

We use a high mutation rate in our approach, as this helps the algorithm avoid local minima. Nevertheless, we do not risk losing good features of the solution space, thanks to the elitist selection applied after crossover or mutation, that is, the fittest solutions are always kept; therefore, if the mutated solutions are less fit than the original solutions, the original solutions are retained.

*4.7. Parameter Selection for Genetic Algorithm.* The way the Genetic Algorithm evolves towards fitter solutions is highly dependent on the parameter selection. In this respect, the percentage of the population on which the crossover and

Table 3: Assignment quality versus population size.

| Nos. of tasks/population | 10 | 20 | 30 | 40 |
|---|---|---|---|---|
| 10 | 10 | 19.7 | 28 | 35.9 |
| 20 | 10 | 19.9 | 28.4 | 36.8 |
| 30 | 10 | 19.9 | 28.7 | 36.5 |
| 40 | 10 | 19.9 | 28.4 | 36.6 |

Table 4: GA parameters.

| | |
|---|---|
| Crossover rate | 0.7 |
| Mutation rate | 0.3 |
| Population size | 30 |
| Max generations | 30 |

mutation steps are implemented is given by the crossover and mutation rate parameters, respectively. As explained in the previous section, a high mutation rate is selected to prevent the algorithm from a too early convergence and falling in local minima. A similar approach is taken in [14, 20]. Moreover, in [14], a similar crossover rate is also selected. Unlike [14] however, we require a lower population size and lower number of generations. This is possibly due to the fact that the elitist selection we implement helps speed up the convergence.

Table 3 shows the impact of the population size on the quality of the assignment solution found (with task quality as single objective) for a different number of tasks considered. We can first see that when the number of tasks increases it becomes more difficult to find an optimal assignment with high quality; this is due to the limited bandwidth and processing constraints in the system. This way, for 10 the tasks considered the assignment found guarantees perfect quality (equal to the number of tasks) while as the task increases the maximum quality attained differs from the perfect quality value.

In addition, we observed that to reach a high-quality solution, it is advisable to use a population with at least the same size than the number of tasks considered. Therefore, for the number of tasks considered in our scenarios, a population size of 30 individuals proves to be suitable. In this respect, we observed that bigger populations cause slow convergence and increased execution cost while small ones tend to evolve to less fit solutions.

In terms of number of iterations, we let the algorithm evolve during 30 generations. This value is experimentally found to be a good tradeoff in terms of achieving a good convergence while still having a reduced execution time.

Table 4 summarizes the parameter selection for the evolution of the Genetic Algorithm. The parameter values selected are also close to the suggested ones in [21].

*4.8. Convergence of Genetic Algorithm.* One way to analyze the convergence of our multiobjective GA is by measuring the area/volume under the Pareto front of solutions in the multiobjective space. The reason is that the minimization of several objectives in our GA translates to Pareto fronts becoming closer to all objective axes, in other words, Pareto fronts with lower areas/volumes underneath.

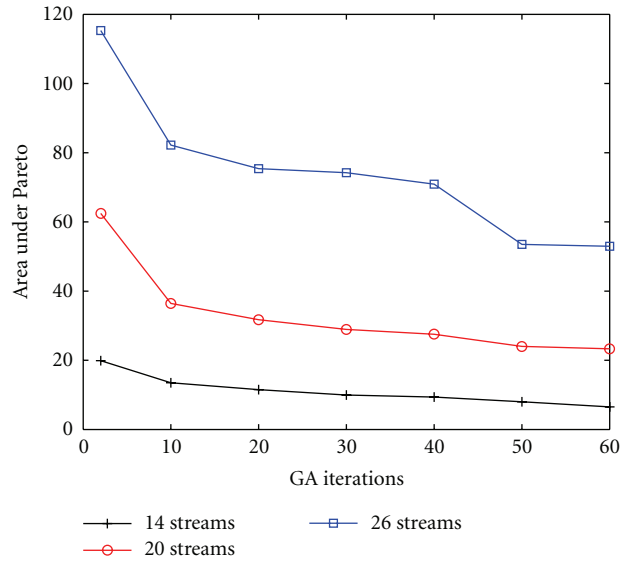



Figure 4: Convergence versus iterations.

Figure 4 shows the evolution of the area under the Pareto front versus the number of iterations for a different number of tasks in the system. In this case, the GA evolves with the objective to minimize both distortion and energy at the client. As we can see with an increasing number of iterations, the area below the Pareto front decreases. This indicates that the Pareto fronts obtained gradually improve and achieve lower values of distortion and energy.

As explained earlier, tenths of iterations are sufficient in our scenario to find good assignment solutions that outperform the reference methods.

*4.9. Execution Cost of Genetic Algorithm.* Our genetic algorithm is an in-house-developed Matlab code and does not form part of the Matlab Optimization toolbox. The code has not been optimized for speed and its average execution time for 10 iterations of the algorithm is in the order of a couple of seconds. In this respect, the computational cost of the reference methods such as MaxQ and Min-Max is almost negligible with respect to GA. However, these methods achieve suboptimal results and are not able to tackle a multiobjective optimization.

Note that genetic algorithms are subject to parallelization, which can speed up its execution considerably. Therefore, a more dedicated and optimized implementation of the algorithm exploiting parallelism would highly reduce its execution time. However, developing such algorithm is out of the scope of this paper. In previous work such as [14], we can observe similar execution times for the GA algorithm and a similar amount of tasks considered. In [22], an optimized implementation of a GA on a SUN 4/490 only requires 1 to 2 s to perform 1000 iterations showing that a dedicated implementation can reduce the execution time considerably. In this respect in our approach, the number of iterations needed

Table 5: Relative execution cost.

| Nos. of tasks/population | 10 | 20 | 30 | 40 |
|---|---|---|---|---|
| 10 | 8 | 9 | 8.5 | 8 |
| 20 | 14 | 15 | 18 | 19 |
| 30 | 20 | 20 | 30 | 32 |
| 40 | 27 | 29 | 40 | 41 |

is in the order of tenths of iterations, which would further reduce the execution time.

In addition, the computational load of the GA is marginal when compared to the high-computational load of any transcoding, decoding operation that takes place in the servers in our scenario. Therefore, the execution of the GA can be placed on such a server with high processing power elements such as a GPU.

Last but not least, in our cloud computing scenario we could expect that new stream processing tasks enter or leave the system not faster than every couple of minutes. Therefore, global or partial recomputations of the stream assignments are not frequently needed.

To give an indication of how the complexity of our algorithm scales, Table 5 shows the relative execution cost versus the number of tasks, and the size of GA population considered. We can see how, as expected, the execution cost increases almost proportionally with the size of the population. The increase with the number of tasks considered is much less noticeable. This has the advantage that scaling up the scenario to a higher number of tasks does not have a big impact on the GA algorithm complexity.

Note also that for large-scale problems, we could address the task scheduling problem in a hierarchical way, that is, tasks can be initially distributed locally among clusters of processing devices and within each cluster; GA can be applied to obtain the optimal assignment. This would limit the complexity increase of the GA optimization.

## 5. Experimental Results

In this section, we compare the performance of the different assignment strategies considered. We first focus on a single objective optimization, namely, quality, where we use the fitness definition in (11). Figure 5 shows the performance of each assignment strategy with respect to the system load (given by an increasing number of SD streams to be displayed at the client's device). We can see that from 4 video streams up to 10, both GA and MaxQ strategies outperform all others while achieving the maximum quality. In this respect, note that the maximum quality equals $N$, the total number of streams, as the maximum quality per stream is 1 (see Table 1). From 10 streams on, it is only the GA that achieves a close to optimal quality. This shows that the higher the load in the system is, the more advanced strategy we need to find a suitable assignment. Moreover, at high load transcoding with slight degradation of the stream quality becomes necessary in order to fit all streams into the available constraints.

We then focus on multiple objectives optimization for a specific system load. We consider the processing of 20 video

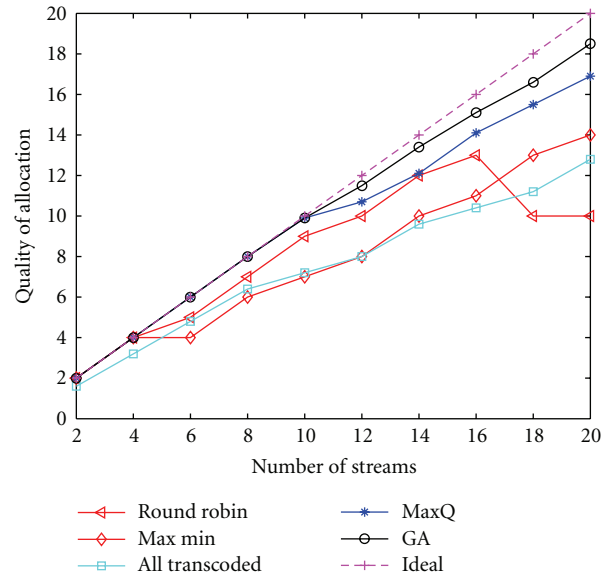



Figure 5: Performance of strategies versus system load.

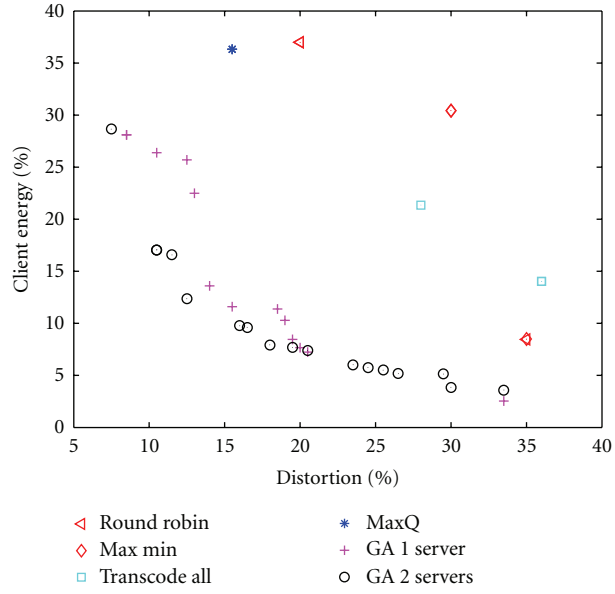



Figure 6: Energy-distortion tradeoffs.

streams of mixed spatial resolutions: CIF, SD, and HD at 30 fps. We compare the different strategies as well as the availability of one single server node with respect to two server nodes where tasks can be migrated to.

Figure 6 shows how the GA clearly outperforms all other strategies when targeting both quality maximization (equivalently distortion minimization) and minimization of energy at the client device. This way, the assignment solutions found by GA incur in lower energy at the client and lower distortion than the solutions found by the other strategies. We can see that in the GA Pareto front, some assignment solutions reach 7% of distortion while the MaxQ Strategy reaches around 15% distortion. Similarly, for the same

Table 6: Assignment strategies for HD/SD/CIF streams.

| | Failed tasks | Decode @client | Decode @server | Temp trans | Spat trans | Temp & spat trans | Dist (%) | Energy (%) |
|---|---|---|---|---|---|---|---|---|
| RR | 0/0/4 | 4/3/2 | 3/4/0 | −/−/− | −/−/− | −/−/− | 20 | 37 |
| MM | 0/0/6 | 7/6/0 | 0/1/0 | −/−/− | −/−/− | −/−/− | 30 | 30.4 |
| TA | 0/0/4 | −/−/− | −/−/− | −/−/− | −/−/− | 7/7/2 | 36 | 14 |
| MaxQ | −/−/− | 2/6/0 | 0/0/1 | 0/0/1 | 0/0/0 | 5/1/4 | 15.5 | 36.3 |
| GA_1 | −/−/− | 4/0/1 | 2/5/0 | 1/2/0 | 0/0/1 | 0/0/4 | 8.5 | 28 |
| GA_2 | −/−/− | 0/0/1 | 3/5/0 | 0/1/0 | 4/0/1 | 0/1/4 | 13 | 22.5 |
| GA_3 | −/−/1 | 0/1/0 | 5/5/0 | 1/0/0 | 1/0/0 | 0/1/5 | 15.5 | 11.6 |

distortion than MaxQ, the GA points lower the energy from 35% to roughly 10%.

Two sets of Pareto solutions are shown for the GA, one corresponding to the use of 1 server and another to the use of 2 servers. Naturally, having two servers available to process, the tasks allow the GA find better assignment solutions. Similarly, for the reference strategies, we show two points (assignment solution) for each strategy displayed. The two points correspond to the use of 1 server and 2 servers respectively, where generally the use of 2 servers achieves lower distortion but also higher energy consumption (as more streams can be processed).

Note that both distortion and energy values are given as percentages from the maximum possible distortion or energy. This way, the maximum energy cost at the client is defined as the cost of processing the decoding tasks of all streams at full spatial and temporal resolution, while the maximum distortion (100%) corresponds to a failed execution for all streams. In general, the relative distortion is defined as

$$\text{Distortion (\%)} = 100\% - \text{Quality (\%)}. \tag{17}$$

This way, from Table 1, successfully processing all $N$ streams at its original spatial-temporal resolution would result in a maximum quality value of $N$ (equivalent to 100% quality or 0% distortion). In a similar way, transcoding the temporal resolution of all $N$ streams would correspond to a total quality of $0.9 * N$ (equivalent to a quality of 90% or distortion of 10%).

We can further analyze the assignment solutions found by the different strategies in Table 6. This table shows some of the assignment solutions of 1 server in Figure 6.

In this table, for the different kind of stream processing assignment, a distinction is made between the different original stream resolutions (HD/SD/CIF). This way, for example, in the assignment of the round robin (RR) strategy 4 CIF streams fail, while 4 HD streams, 3 SD, and 2 CIF are successfully decoded at the client, and 3 HD and 4 SD streams are decoded at the server. Finally, no streams are transcoded in this strategy. By analyzing Figure 6 and Table 6, we can see that strategies such as round-robin (RR) and max-min (MM) cannot find an assignment with a good tradeoff in terms of quality and energy cost. The reason is that none of these strategies considers either bandwidth constraints or transcoding. In this example, both round-robin and max-min succeed in distributing the load evenly among nodes and meet the processing constraints. However, decoding streams at the server nodes generates high bandwidth demands towards the client. This exceeds the bandwidth availability and causes failed processed streams (in particular the high bandwidth HD streams). Therefore in this case the use of 2 servers further degrades the performance as more streams are processed at the server's side, and this aggravates the bandwidth demand. This way, under high load some stream transcoding would be required to fit into both available processing and bandwidth constraints.

However, transcoding all streams at the server nodes (TA strategy) is neither an optimal assignment as we enforce the quality degradation of all streams. Moreover, transcoding tasks are processing intensive and exceed the processing capacity at the servers resulting in some failed stream processing. In this case, the use of 2 servers increases the overall processing capacity but the assignment remains quite suboptimal in terms of distortion and energy.

In contrast, the MaxQ strategy succeeds in finding an assignment with low distortion value (15%). However, its energy cost is relatively high (36%). In Table 6, we can see that all 20 streams are successfully processed, out of which 10 are transcoded temporally and spatially while 8 are decoded at the client device at full resolution. In this example, the processing capacities with one server are enough for the amount of transcoding involved, while the main limiting factor remains the bandwidth.

It is finally the GA that outperforms all strategies by addressing both objectives and finding a set of assignment solutions, that is, Pareto optimal in both senses. Moreover, having 2 server nodes available for processing allows the GA to find even better tradeoffs (lower Pareto curve) in terms of quality and energy. Note also that the set of GA solutions offers an energy range from 30% to 10% for low distortion values. This way, we can reduce the energy at the client by factor 3 by trading off some stream quality. This offers the flexibility to choose between different operating points at grid level according to how scarce the processing power and energy at the client is.

In Table 6, we only show a few of the set of Pareto optimal solutions found by the GA. We can see how GA_1 assignment solution reaches the lowest distortion (8.5%) at 20% lower energy cost than MaxQ strategy. This assignment distributes more evenly the streams between client and server and chooses to transcode temporally and spatially only 4 streams of HD resolution. The GA_2 assignment provides a tradeoff

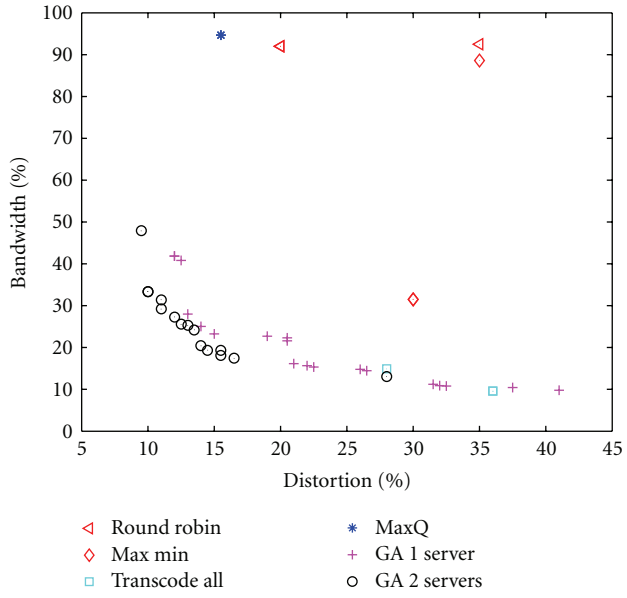



Figure 7: Bandwidth-distortion tradeoffs.

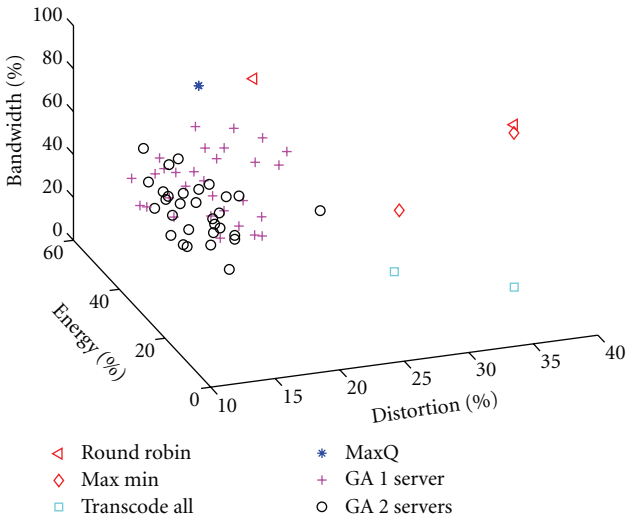



Figure 8: Pareto points in 3D space.

of slightly higher distortion (13%) but an energy cost 40% lower than that provided by MaxQ. Finally, GA_3 assignment provides similar distortion than MaxQ (15.5%) but it further reduces the energy consumption, demanding almost 70% less energy than MaxQ.

Figure 7 shows a similar comparison with respect to the bandwidth usage and distortion incurred by the different assignment solutions. In this case, the GA targets optimization of both distortion and bandwidth objectives and clearly outperforms all other strategies as shown by the lower distortion and bandwidth values obtained in the GA Pareto front. We can see in this respect how the MaxQ solution has a high bandwidth cost (close to 100%) while the GA solutions in the Pareto front require for the same distortion barely 20% of the maximum bandwidth. In addition, considering two

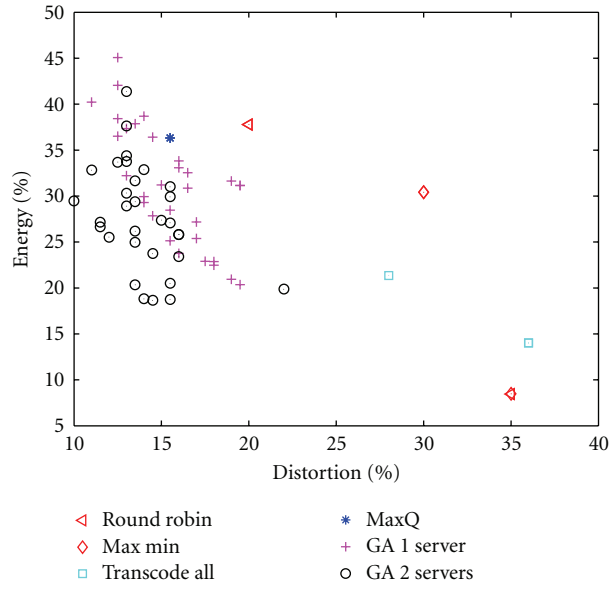



Figure 9: Projection onto energy-distortion space.

servers further allows a distortion and bandwidth reduction, possibly due to the fact that more processing power is available for transcoding the highly demanding HD streams.

Finally, Figure 8 shows the optimality of the different assignments in the three-dimensional space of distortion, bandwidth usage, and client's energy cost. The GA targets the minimization of these three objectives and the Pareto front becomes a Pareto surface.

As in the previous figures, the assignments found for 1 and 2 servers are displayed. Once again, we can see that the assignments found by the GA outperform all other strategies in terms of distortion, energy and bandwidth while at the same time, it provides a good tradeoff for all three objectives. Indeed, we can see that the GA solution points concentrate around lower distortion values (especially those corresponding to use of 2 servers), lower bandwidth, and lower energy values. For the sake of clarity, in Figure 9, we show the projection of Figure 8 on the two-dimensional space of energy and distortion where the better results of GA can be seen clearly.

In practice, both processing and bandwidth constraints may vary over time. Therefore, we may require a new stream assignment to fit the new constraints. One possible way to tackle this is by simply rerunning the assignment strategy. However, as the GA strategy already produces a set of assignment solutions with different energy-distortion-bandwidth tradeoffs, another possibility is to simply choose from the set of solutions a different operating point (assignment solution) that satisfies the new constraints. This way, if the current assignment solution requires a processing of 50% at the client's side, switching to a new assignment with 30% processing may be suitable for a more overloaded client device. We can also cope with variations in bandwidth or processing constraints by targeting more limiting constraints, for instance, 80% of the maximum bandwidth and maximum processing power. By doing so, the assignment is slightly overdimensioned and can cope with variations of up

to 20–25% above the current constraints. This would also help avoid too frequent task migrations in the system.

## 6. Conclusion

We have presented an evolutionary-based strategy for stream processing assignment in a client-cloud multimedia system where multiple heterogeneous devices are considered. In this context, we not only decide on which node each stream is assigned but we also consider the possibility of stream transcoding to a lower temporal or spatial resolution. This extends the system capacity at the cost of smooth quality degradation in the task execution.

Moreover, both processing capacities in the nodes and bandwidth availability are taken into consideration. The proposed strategy is highly flexible and can target multiple objectives simultaneously. It outperforms all other considered strategies while providing a wide range of tradeoffs in the assignment solutions.

## Acknowledgment

The authors would like to thank Yiannis Iosifidis for his insights on genetic algorithms.

## References

[1] Y.-W. Zhong and J.-G. Yang, "A hybrid genetic algorithm with Lamarckian individual learning for tasks scheduling," in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics (SMC '04)*, pp. 3343–3348, October 2004.

[2] K. Dahal, A. Hossain, B. Varghese, A. Abraham, F. Xhafa, and A. Daradoumis, "Scheduling in multiprocessor system using genetic algorithms," in *Proceedings of the 7th IEEE Computer Information Systems and Industrial Management Applications (CISIM '08)*, pp. 281–286, June 2008.

[3] M. R. Miryani and M. Naghibzadeh, "Hard real-time multi-objective scheduling in heterogeneous systems using genetic algorithms," in *Proceedings of the 14th International CSI Computer Conference (CSICC '09)*, pp. 437–445, October 2009.

[4] J. Carretero, F. Xhafa, and A. Abraham, "Genetic algorithm based schedulers for grid computing systems," *International Journal of Innovative Computing, Information and Control*, vol. 3, no. 6, pp. 1053–1071, 2007.

[5] R. Entezari-Maleki and A. Movaghar, "A genetic algorithm to increase the throughput of the computational grids," *International Journal of Grid and Distributed Computing*, vol. 4, no. 2, 2011.

[6] J. Liu, L. Chen, Y. Dun, L. Liu, and G. Dong, "The research of ant colony and genetic algorithm in grid Task scheduling," in *Proceedings of the International Conference on MultiMedia and Information Technology, (MMIT '08)*, pp. 47–49, December 2008.

[7] A. Folling, C. Grimme, J. Lepping, and A. Papaspyrou, "Robust load delegation in service grid environments," *IEEE Transactions on Parallel and Distributed Systems*, vol. 21, no. 9, pp. 1304–1316, 2010.

[8] K.-M. Yu and C.-K. Chen, "An evolution-based dynamic scheduling algorithm in grid computing environment," in *Proceedings of the 8th International Conference on Intelligent Systems Design and Applications (ISDA '08)*, pp. 450–455, November 2008.

[9] A. Michalas and M. Louta, "Adaptive task scheduling in grid computing environments," in *Proceedings of the 4th International Workshop on Semantic Media Adaptation and Personalization (SMAP '09)*, pp. 115–120, December 2009.

[10] X. Yang, J. Zeng, and J. Liang, "Apply MGA to multi-objective flexible job shop scheduling problem," in *Proceedings of the International Conference on Information Management, Innovation Management and Industrial Engineering (ICIII '09)*, pp. 436–439, December 2009.

[11] M. Basseur, F. Seynhaeve, and E.-G. Talbi, "Path relinking in Pareto multi-objective genetic algorithms," in *Proceedings of the 3rd International Conference on Evolutionary Multi-Criterion Optimization (EMO '05)*, pp. 120–134, March 2005.

[12] P. Chitra, S. Revathi, P. Venkatesh, and R. Rajaram, "Evolutionary algorithmic approaches for solving three objectives task scheduling problem on heterogeneous systems," in *Proceedings of the IEEE 2nd International Advance Computing Conference (IACC '10)*, pp. 38–43, February 2010.

[13] R. P. Dick and N. K. Jha, "MOGAC: a multiobjective genetic algorithm for hardware-software cosynthesis of distributed embedded systems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 17, no. 10, pp. 920–935, 1998.

[14] M. Camelo, Y. Donoso, and H. Castro, "MAGS—an approach using multi-objective evolutionary algorithms for grid task scheduling," *International Journal of Applied Mathematics and Informatics*, vol. 5, no. 2, 2011.

[15] F. S. Kazemi and R. Tavakkoli-Moghaddam, "Soving a multi-objective multi-mode resource-constrained project scheduling problem with particle swarm optimization," *International Journal of Academic Research*, vol. 3, no. 1, 2011.

[16] Y. Hu and B. Gong, "Multi-objective optimization approaches using a CE-ACO inspired strategy to improve grid jobs scheduling," in *Proceedings of the 4th ChinaGrid Annual Conference (ChinaGrid '09)*, pp. 53–58, August 2009.

[17] M. Maheswaran, S. Ali, H. J. Siegel, D. Hensgen, and R. Freund, "Dynamic mapping of a class of independent tasks onto heterogeneous computing systems," in *Proceedings of the 8th IEEE Heterogeneous Computing Workshop (HCW '99)*, pp. 30–44, San Juan, Puerto Rico, April 1999.

[18] N. Srinivas and K. Deb, "Multi-objective optimization using non-dominated sorting in genetic algorithms," *Evolution Computing*, vol. 2, no. 3, pp. 221–248, 1994.

[19] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 257–271, 1999.

[20] Y. Iosifidis, A. Mallik, S. Mamagkakis et al., "A framework for automatic parallelization, static and dynamic memory optimization in MPSoC platforms," in *Proceedings of the 47th Design Automation Conference, (DAC '10)*, pp. 549–554, June 2010.

[21] A. Y. Zomaya, C. Ward, and B. Macey, "Genetic scheduling for parallel processor systems: Comparative studies and performance issues," *IEEE Transactions on Parallel and Distributed Systems*, vol. 10, no. 8, pp. 795–812, 1999.

[22] S. H. Hou, N. Ansari, and H. Ren, "Genetic algorithm for multiprocessor scheduling," *IEEE Transactions on Parallel and Distributed Systems*, vol. 5, no. 2, pp. 113–120, 1994.

Hindawi Publishing Corporation
International Journal of Digital Multimedia Broadcasting
Volume 2012, Article ID 935724, 11 pages
doi:10.1155/2012/935724

*Research Article*

# Towards an Automatic Parameter-Tuning Framework for Cost Optimization on Video Encoding Cloud

**Xiaowei Li, Yi Cui, and Yuan Xue**

*Department of Electrical Engineering and Computer Science, Vanderbilt University, Nashville, TN 37235, USA*

Correspondence should be addressed to Xiaowei Li, xiaowei.li@vanderbilt.edu

Received 15 May 2011; Revised 27 September 2011; Accepted 11 October 2011

Academic Editor: Yifeng He



The emergence of cloud encoding services facilitates many content owners, such as the online video vendors, to transcode their digital videos without infrastructure setup. Such service provider charges the customers only based on their resource consumption. For both the service provider and customers, lowering the resource consumption while maintaining the quality is valuable and desirable. Thus, to choose a cost-effective encoding parameter, configuration is essential and challenging due to the tradeoff between bitrate, encoding speed, and resulting quality. In this paper, we explore the feasibility of an automatic parameter-tuning framework, based on which the above objective can be achieved. We introduce a simple service model, which combines the bitrate and encoding speed into a single value: encoding cost. Then, we conduct an empirical study to examine the relationship between the encoding cost and various parameter settings. Our experiment is based on the one-pass Constant Rate Factor method in x264, which can achieve relatively stable perceptive quality, and we vary each parameter we choose to observe how the encoding cost changes. The experiment results show that the tested parameters can be independently tuned to minimize the encoding cost, which makes the automatic parameter-tuning framework feasible and promising for optimizing the cost on video encoding cloud.

## 1. Introduction

The consensus of both academia and industry has been reached that cloud computing will be the next revolutionary progress in information technology, in which dynamically scalable and virtualized computation resources are provided as service over Internet. It has drawn extensive attention from IT giants, such as Amazon (EC2, Elastic Computing Cloud; S3, Simple Storage Service) [1], Google (GAE, Google Application Engine) [2], and Microsoft (Azure) [3]. Customers are charged based on the amount of computation resources they actually have consumed, including CPU time, memory, storage, and transferred bytes. For example, Amazon EC2 charges $0.125 per hour for standard on-demand CPU instances running Windows and $1.20 for extra-large high-CPU instances as for now [1].

Online video vendors like YouTube [4], Youku [5], and Tudou [6] have a huge number of videos of various formats uploaded by users every day, which need to be transcoded (format converted) into their desired intermediate formats, such as .flv, for storage and replay. Such transcoding work is computing-intensive and time-consuming. It also requires storage space and bandwidth for transfer, which pose extremely high demands on hardware resources. Many content owners (libraries, traditional media production units, and governments) have the need to digitize and distribute their contents, but no technical resources/expertise to do so. With the introduction of cloud computing, a new type of Internet service termed encoding cloud emerges, which builds on the cloud infrastructure (e.g., Amazon EC2) to provide value-added video encoding service to meet the aforementioned demands. A successful example is Flix Cloud [7], which combines On2's Flix Engine encoding software with Zencoder's web-based application to provide scalable online encoding service based on Amazon EC2. Another example is http://www.encoding.com/, which offers a similar web-based "pay-as-you-go" service.

For both encoding service providers and customers, a good trade-off between quality and speed is desirable for cost saving purposes. The difficulty originates from the inherent complexity of modern video encoding technologies, most of which inherit from the block-based hybrid coding

design. For example, x264, a widely adopted video encoding engine, exposes more than 100 parameters [8] for its users to balance a long list of factors including encoding speed, bitrate, objective/subjective quality, decoding buffer, and so forth. Evidently, it would be prohibitively expensive for an operator to figure out the most economically efficient "encoding recipes" for thousands of uploaded videos every day. Nevertheless, any endeavor towards this goal is imperative and essential, since the cost saving on the cloud resource consumption will simply add to the profits, whose amount would be quite considerable given the sheer volume of video being encoded. To this end, we believe that an automated mechanism for encoding parameter selection and tuning would be very valuable and timely.

In this paper, we commence our exploration towards this goal. To the best of our knowledge, there is no prior work on this front. Our study chooses H.264 as the targeted video coding standard, and specifically x264 as the targeted encoding software. The rationale of our choice is delayed to Section 2. Our empirical study makes the following important findings.

(i) Modern video encoding technology, represented by x264, is able to precisely and consistently achieve any specified numerical target on subjective/objective quality metric, such as PSNR (peak signal-to-noise ratio). This is a must-have property for any cost-optimization framework desiring to meet stringent encoding quality constraint.

(ii) While observing the quality constraint, we find that many key video encoding parameters (e.g., B-frame) are shown to be able to be independently tuned to adjust the encoding cost.

(iii) There are also strong indications that the impact of each parameter to the final encoding cost can be specified in a weighted sum fashion, in which the weight stays invariant to external factors such as cost coefficients. Combined with previous two observations, it suggests the feasibility of an automatic tuning framework over decomposable parameters, which can quickly find the optimal operation point and predict the encoding cost, if certain knowledge on encoding complexity of the video itself is available a priori.

The rest of this paper is organized as follows. Section 2 reasons our selection on x264 and reveals the internal relationship among bitrate, quality, and encoding speed. Section 3 defines a simple service model under the cloud computing environment. Section 4 presents the experimental results on key parameters in x264, such as B-frame, Ref, Subme, and Trellis. By examining critical observations made here, Section 5 discusses promises and hurdles towards an automatic cost tuning framework. Some related work are described in Section 6. Section 7 concludes this paper.

## 2. Related Work

Some research work [9–11] propose various algorithms, including mode ranking and learning theoretic, to optimize the R-D (rate-distortion) performance while constraining the computational complexity. Although they also address the ternary relationship between bitrate, encoding time, and quality, their approaches aim at improving the encoding technologies at the macroblock level. References [12, 13] also address the issues associated with handheld devices and power-constrained environments. The most related work to ours are the algorithms proposed by Vanam et al. [14, 15], which are employed to select the parameter settings that have comparable performances with those chosen by exhaustive search using a much fewer number of trials. Rhee et al. [16] propose a real-time encoder, which dynamically adjusts the predefined complexity level to meet the target time budget while maintaining the encoding quality and efficiency.

Both our objective and methodology are different from theirs. First, we are trying to establish a parameter-tuning framework, which can be used for optimizing the encoding cost (or resource consumption) based on a specific service model, while they aim at optimizing the R-D performance. Second, we do not touch the internal of the encoding software, but regard it as a black-box system, whose input is the encoding parameter settings and the original video. We would like to examine the correlation between different parameter settings and encoding performances regardless of the tested video, essentially how the encoding parameters affect the behavior of the encoding software. The parameter-tuning framework is established on the empirical study over a large number of videos. Given a specific video whose encoding complexity can be estimated, an optimal parameter setting which minimizes the encoding cost with satisfiable quality will be generated.

## 3. Encoding in x264

*3.1. Rationale.* In this study, we choose H.264 as the video coding standard and use x264 as the encoding software. Our choice of H.264 roots on the simple fact that it has been recognized as the most pervasive video coding standard for web videos with the support of Flash Player 10 by its outstanding compression and efficiency. Currently, the encoding cloud services have integrated a large number of encoding/transcoding software, either open-source or commercial, including On2 Flix [7], Mencoder [17], and Zencoder [18]. Among burgeoning H.264 codec, x264 [19] is one of the best performers (supported by MSU Annual H.264 Codec Comparison Report [20]) and adopted by many widely popular applications including ffmpeg [21], Mencoder, and similar x264 implements almost all of the H.264 features, which means that users have a pretty large parameter configuration space to accommodate their specific encoding requirements, a particular merit of x264. Furthermore, experiments conducted by YUV Soft [22] have shown that using x264 could achieve either excellent compression quality or extremely fast encoding speed with great flexibility, depending on the parameter settings.

*3.2. Encoding Model.* We model a video for encoding using two parameters:

(i) $N$: the number of frames;

(ii) $F$: the frame rate (frame per second), which is the number of frames played in one second.

We use three metrics for evaluating the encoding configuration under H.264.

(i) $V$: the encoding speed (second/frame), which is the time required to encode one frame. We use encoding rate and encoding speed interchangeably in this paper;

(ii) $R$: the bitrate after compression with container overhead (kbps), which implies the final file size and the compression ratio;

(iii) *PSNR*: the peak signal-to-noise ratio, which represents the resulting objective quality. There are two variations, average PSNR and global PSNR. Usually global PSNR is a little lower than average PSNR and we choose average PSNR in this paper.

All the above metrics can be easily obtained from the output statistics of x264.

*3.3. Rate Control Modes in x264.* Modern video encoding software, combined with the flexibility promised by the encoding standard such as H.264, offers handy and functional rate control methods to determine how bits are allocated within streams and tune the delicate balance among bitrate ($R$), encoding speed ($V$), and objective quality (*PSNR*). There are four rate control modes in x264, which are used most frequently, namely, CQP (Constant Quantization Parameter), CRF (Constant Rate Factor), ABR (Average Bit Rate), and two-pass VBR (Variable Bit Rate).

CQP mode specifies the same quantization values for all P-frames, meanwhile fixing the relative ratios (i.e., ipratio and pbratio) of quantization values for I-frames and B-frames in the same GOP (group of pictures). CQP mode disables adaptive quantization, a technique aiming to keep constant visual quality by imposing larger quantization values on high-complexity scenes and lower ones on low-motion scenes, based on the assumption that humans are less perceptive to high-motion scenes.

CRF mode targets a certain level of "quality" by adaptively adjusting quantization values for frames with different complexity by setting a constant rate factor. A smaller rate factor means higher quality and higher bitrate. A constant rate factor aims at consistent visual quality, while it may sacrifice some objective quality (PSNR). Generally, CRF mode produces smaller file size than CQP mode and is the most recommended one-pass mode if the quality concern overrides the strictness of the desired bitrate.

ABR and VBR modes focus on achieving the desired bitrate rather than the objective or subjective quality. Reference [23] shows that ABR mode achieves lower PSNR than CRF mode and cannot achieve the exact final file size as specified in one pass. On the same note, two-pass VBR mode gives more accurate file size with possibly the highest quality. However, much more time is needed since the encoding is done practically twice, where the second pass achieves much more accurate bitrate by learning from statistics resulted from the first pass.

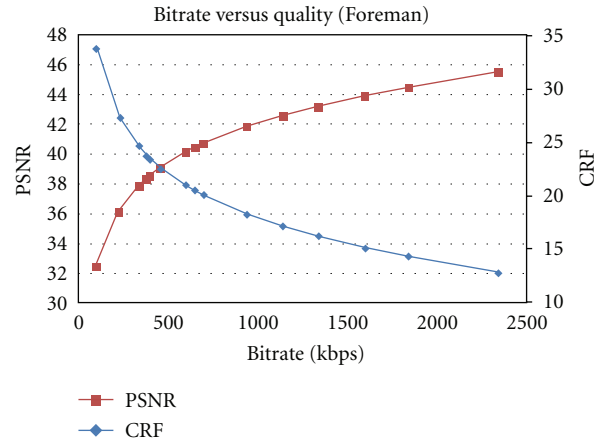



Figure 1: Bitrate versus PSNR and CRF (Foreman).

*3.4. Bitrate, Encoding Speed, and Quality.* In this section, we obtain a basic understanding of the relationship between the above three evaluation metrics. We choose the ABR mode using which we can target different bitrates and "foreman" as the example video file. Figure 1 shows the relationship between the bitrate and the resulting PSNR. We can see that the larger the bitrate, the better the quality, which is expected, although the quality improvement is becoming smaller with the escalation of bitrate. We also plot the final rate factor with varying bitrates, which is the average quantization value for all types of frames. We can see that the final rate factor is in near reverse proportional with PSNR. Since the rate factor is the major tuning parameter in CRF mode, using CRF mode can target relatively accurate video quality. Figure 2 shows the relationship between the bitrate and the encoding speed. We can see that the larger the bitrate, the longer it requires to encode a frame, since more bits are allocated. In summary, the better video quality requires larger bitrate (final file size) and longer encoding time.

## 4. Service Model

It is clear that changing encoding parameters simultaneously affects bitrate, encoding speed, and video quality. Thus the optimal selection of encoding parameters becomes a multiobjective optimization problem, where the tradeoffs and constraints among these objectives (e.g., bitrate, encoding speed, and video quality) are dependent on the user's preference and cost structure. Cloud service providers charge the customers based on the resource consumption. They naturally provide a unified cost structure, in which CPU usage time, memory, bandwidth, and storage expenses are represented by singular monetary value and combined as the final charges. The encoding service consumes all major resource types, including CPU usage time, memory, storage

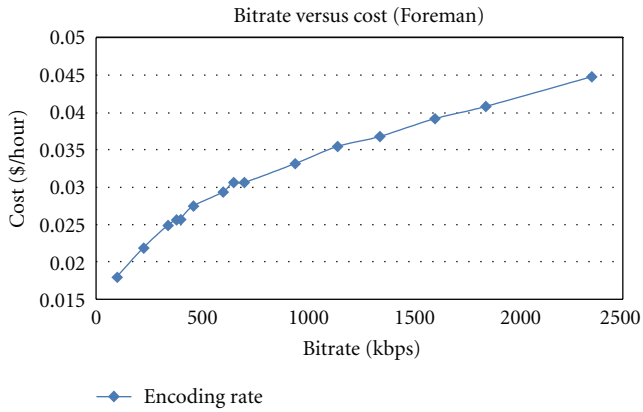



Figure 2: Bitrate versus Encoding speed (Foreman).

space, and bandwidth for data transfer. Here we omit the memory usage in our model, considering the facts that an average modern computer in the cloud (e.g., Amazon EC2 instance) has abundant memory for encoding software and memory consumption alone is not charged by current cloud service vendors. Obviously, the encoding process and the encoding results would affect the usage of various types of resource. Among the remaining three, CPU usage is only related with the encoding of the video, which is a one-time job. The CPU usage can be represented using the encoding time ($T$), which is calculated as

$$T = N \times V \text{ (second)}. \tag{1}$$

Recall that $N$ is the number of frames and $V$ is the encoding speed as defined in Section 2. The storage and bandwidth (file transfer) usage depends on the compressed file size ($S$), calculated as

$$S = \frac{N}{F} \times \frac{R}{8} \text{ (byte)}. \tag{2}$$

Recall that $R$ is the bitrate after compression and $F$ is the frame rate (frame per second) as defined in Section 2. Different from CPU usage, the storage and transfer costs of a video can be recurring, and dependent on different encoding service models. We define the cloud encoding cost as COST and merge the encoding time, the storage, and the bandwidth usage into a singular monetary value: operational cost of encoding (briefly encoding cost) in the cloud: COST ($)

$$\text{COST} = \alpha \times T + \beta \times S = N \times \left(\alpha V + \frac{\beta R}{8F}\right), \tag{3}$$

where $\alpha$ and $\beta$ are cost coefficients related with CPU usage and storage/bandwidth resources, respectively. Since different videos have different number of frames, we could normalize the above cost by eliminating $N$. We obtain the following normalized cost (COST′, $ per hour), which implies the cost to encode an hour long of this specific video

$$\text{COST}' = \text{COST} \times \frac{3600}{N/F} = 3600 \times \left(\alpha V F + \frac{\beta R}{8}\right). \tag{4}$$

The new cost value, irrelevant with the number of frames, is varied with different videos, since the encoding complexity of a specific video is different from others. The cost coefficients $\alpha$ can be easily set as the current hourly CPU price. For example, we use $0.34/hour here, which is the price for large standard on-demand instances of Amazon EC2. On the other hand, $\beta$ is dependent on the service model. For example, in the encoding-only service model, the Cloud platform is only used for the computationally extensive encoding tasks. It downloads the encoded video to its local hosting platform for long-term video storage and video distribution. In this case, the storage cost is limited to short term and there is only one-time transfer cost. For example, for transfer cost, we use the Amazon EC2's receding staircase cost starting at $0.15/GB; for the storage cost, we use $0.15/GB/month of Amazon S3 for two weeks. In the full-fledge video service model, the Cloud platform is responsible for long-term storage and distribution in addition to encoding. In this case, the storage cost depends on the storage time and the transfer cost depends on the amount of video access. Since the lifetime of the video to remain in the cloud and its popularity (hence its distribution cost) will be hard to determine, albeit a separate issue. The difficulty of predicting the above values depends on the time horizon. Nevertheless, some simple intuition developed later in this paper can be useful when uncertainty has to be dealt with.

In this work, we consider the encoding-only service model and set the goal of encoding as to maintaining the encoded quality to be acceptable by users through service-level agreement (SLA). In this way, factors other than the encoding aspect of a video (such as its popularity) are excluded.

## 5. Experiment Study

In the following encoding experiments, we examine the effects of parameter settings on encoding time, bitrate, and objective quality (PSNR), the collection of which determines the final cost. The test sequences we utilized are downloaded from [24], a well-recognized video test set. They are all in CIF format, 25 frames/second, and resolution of 352 × 288. We have tested 12 videos in total, including Bus, Coastguard, Container, Foreman, Flower, Hall, Mobile, News, Silent, Stefan, Tempete, and Waterfall. In order to minimize the stochastic errors, every set of experiments is performed five times and the results are averaged.

To align with the necessity to maintain certain acceptable level of quality, we must choose rate control methods best at controlling quality, which leaves us CQP and CRF. If purely measured by objective quality, that is, PSNR, CQP is a better candidate than CRF. But CRF brings about significant benefits, which are (1) saving more bitrate than CQP and (2) maintaining constant subjective quality, a value much appreciated by end users. As such, we believe that CRF will be commonly preferred by encoding service providers as the primary rate control method, which is also our choice in this study.

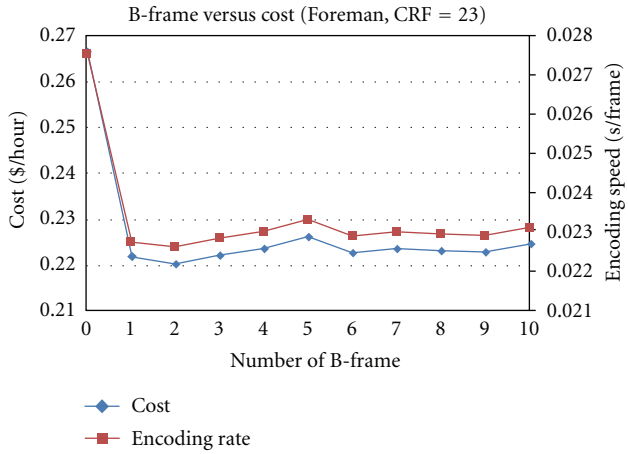



Figure 3: Cost and encoding speed with varying B-frame number (Foreman, crf = 23).

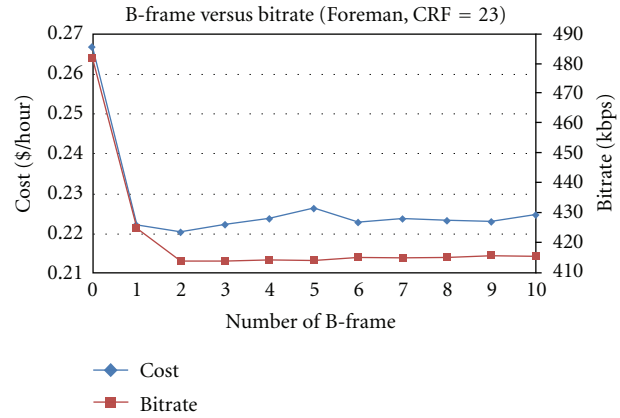



Figure 4: Bitrate and cost with varying B-frame number (Foreman, crf = 23).

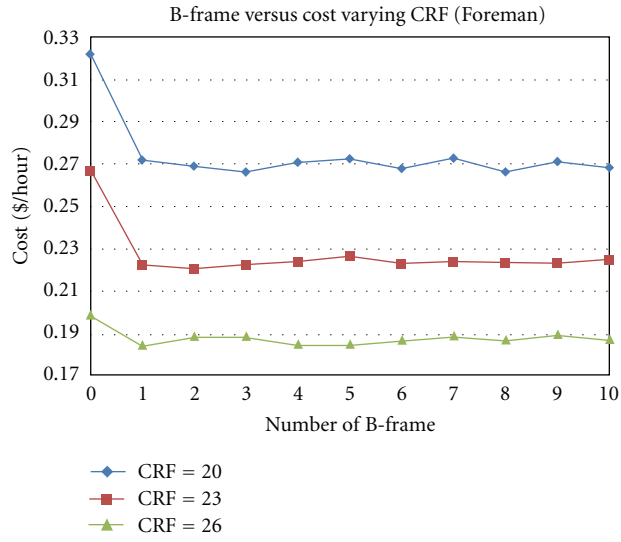



Figure 5: Encoding cost with varying B-frame number under different CRF settings (Foreman).

The excellent performance of x264 is attributed to several factors, including motion estimation, frame-type decision, macroblock mode decision, and quantization. We first vary one parameter while keeping all others default, and then test multiple parameters to examine the composite effect. We note that what will frequently show up in the following figures is a parameter which is also called *CRF*. It sets the target to the CRF method to achieve the perceptual quality effect as if the same quantization value (default value is 23 in x264) is applied to the encoded video.

In the following experiments, if not clearly stated, all the test sequences exhibit similar curves as the example Foreman we will use below, under three quality levels, for concise purpose.

*5.1. B-Frame.* B-frame setting determines the maximum number of consecutive B-frames we can use. B-frame, different from P-frame, can use motion prediction from future frames, which enhances the compression efficiency.

Figure 3 shows the encoding cost and rate with varying B-frame numbers. We can see that the use of B-frame dramatically decreases the encoding cost, which fluctuates a little bit with higher B-frame setting. The frame-type decision algorithm in x264 for replacing a P-frame with B-frame is fast and flexible. Notice that the cost curve closely matches the encoding speed, which becomes the dominant factor using our chosen cost coefficients $\alpha$ and $\beta$, and the minimum cost may be achieved at different points if coefficients vary.

Figure 4 shows the bitrate and encoding cost with varying B-frame numbers. We can see that the resulting bitrate drops with the use of B-frame and down to stable level at the point of two B-frames. We speculate that the most effective number of B-frames for Foreman is 2, which is supported by the consecutive B-frame distributions extracted from x264 output statistics. The distribution of consecutive B-frames for Foreman changes from {13.4%, 86.6%} to {13.1%, 69.8%, 17.1%} if we vary the B-frame number from 1 to 2, which means with one B-frame setting, 13.4% of frames use no consecutive B-frame and 86.6% use one B-frame, while with two B-frame setting, 17.1% of frames use two consecutive B-frames. If we set a cutoff threshold for the cumulative distribution, say 90%, we can see the B-frame number is set to 2, which matches our observation. We also did the same experiments over other videos and conclude that it is appropriate for most videos to use 1 up to 3 B-frames to save significant bitrates as well as encoding time and cost while higher B-frame settings contribute little.

Figure 5 shows the encoding cost with different number of B-frames under varying CRF settings. We can see expected results that the encoding cost increases with CRF values and has marginal variations with higher B-frame numbers. Figure 6 shows the averaged encoding cost using 2 B-frames under different quality levels (CRF value) for all videos. We can see the cost rankings of videos under a specific CRF value are consistent for all the three quality levels (except Hall with CRF = 20). We speculate that it is the encoding complexity, the inherent characteristic of video, that leads to this phenomenon.

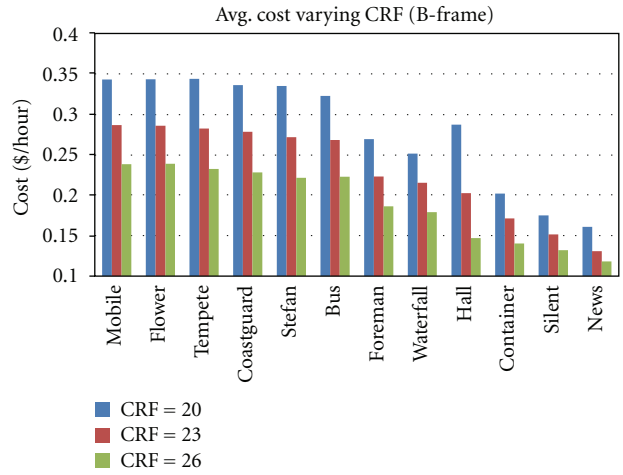



Figure 6: Averaged encoding cost for all videos.

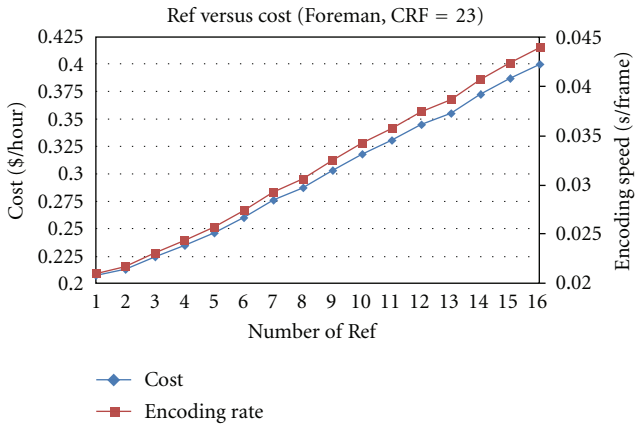



Figure 7: Encoding cost and rate with varying Ref number (Foreman, crf = 23).

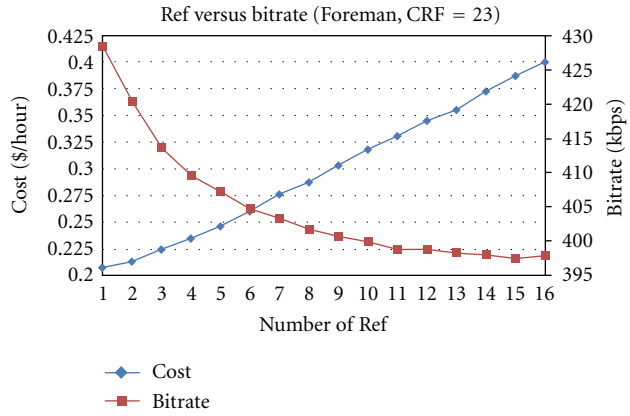



Figure 8: Bitrate and cost with varying Ref number (Foreman, crf = 23).

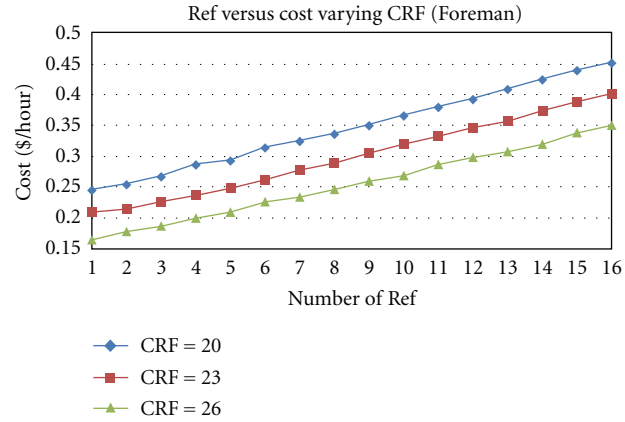



Figure 9: Encoding cost with varying Ref number under different CRF settings (Foreman).

*5.2. Ref.* Ref is a parameter referring to the number of previous frames that a P-frame can refer to, ranging from 1 to 16. Conceptually, each increment of Ref will bring about constant speed loss, and on the flip side, the benefit of lower bitrate, especially for highly animated videos. On the other hand, the number is dependent on both DPB (Decoded Picture Buffer) confined by ITU for specific playback devices and the preference over speed or bitrate.

Figures 7 and 8 show the encoding cost, rate, and bitrate with varying Ref numbers. As expected, a larger Ref number leads to increasing encoding time and cost, as well as decreasing bitrate. The encoding cost is mostly influenced by speed loss, and the deviation resulting from reduced bitrate is not prominent. Compared to P-frames, B-frames usually use one or two fewer reference frames. If we examine the bitrate more closely, the decreasing gradient and the relatively "stable" point for each video are different from each other.

There are two explanations. One is our choice of cost coefficients, which puts more weight on encoding speed (CPU cost) over bitrate (storage and bandwidth costs). On the same note, it also suggests that the speed loss overshadows the bitrate benefit. The encoding time increases by about two times when Ref varies from 1 to 16, while the bitrate only drops around 30 kbps from original 430 kbps. For some videos, the bitrate variance is within 10 kbps. As such, we may suggest using as few Ref as possible for most videos, except some with high motion or complexity.

Then we vary CRF and get expected result in Figure 9 that the encoding cost increases with the quality improvement. Figure 10 displays the average cost (in most cases also the median value due to approximately linear increase) when varying CRF for all videos. Like Figure 6, the ranking is consistent with Hall being the only exception.

*5.3. Subme.* Subme controls the subpixel estimation complexity. Since H.264 has a wide range of macroblock partitions for both intra- and interpredictions, mode decision is very time-consuming. Subme levels from 0 to 5 increase the level of refinement, while levels from 6 to 9 enable a hybrid decision of SATD (sum of absolute transformed differences) and RDO (rate distortion optimization) for corresponding frame types. We do not consider level 10 since

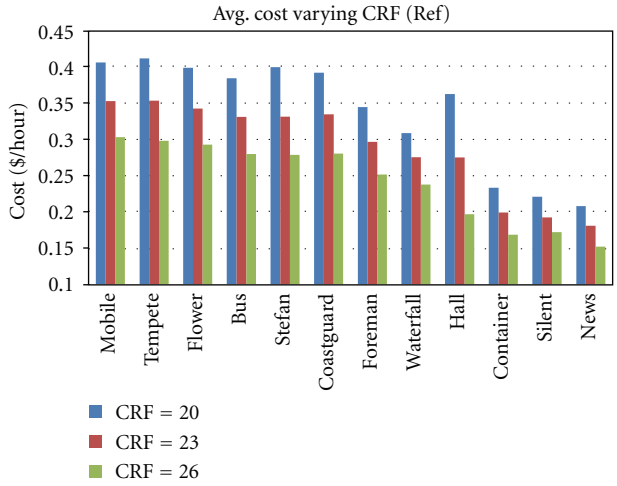



Figure 10: Averaged encoding cost for all videos.

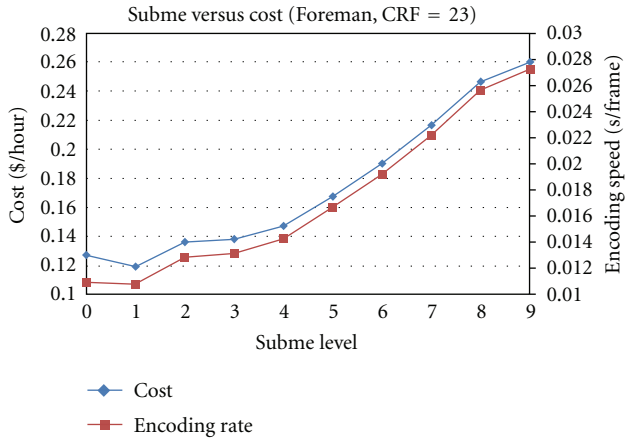



Figure 11: Encoding cost and rate with increasing Subme levels (Foreman, crf = 23).

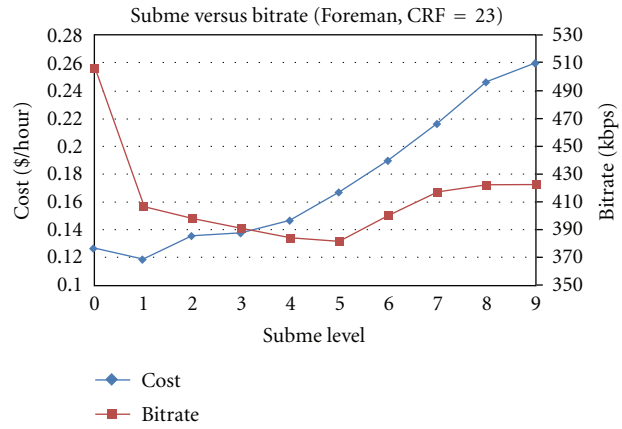



Figure 12: Bitrate and cost with increasing Subme levels (Foreman, crf = 23).

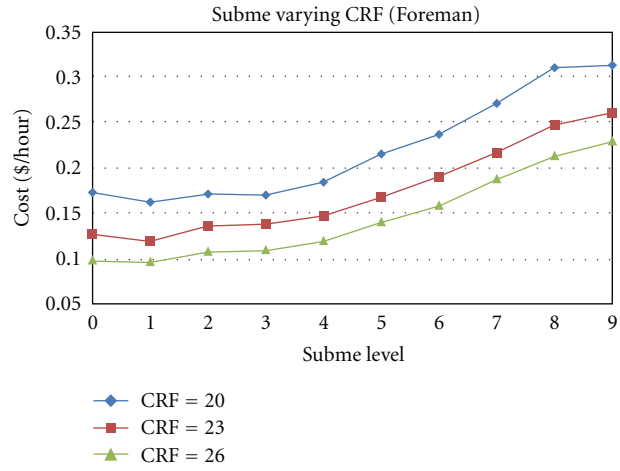



Figure 13: Encoding cost with increasing Subme levels under different CRF settings (Foreman).

it requires trellis option to be enabled, which only applies to extreme cases. Intuitively, higher Subme level leads to better compression efficiency at the price of encoding speed loss.

Figures 11 and 12 show the encoding cost, rate, and bitrate with increasing Subme levels. We can see that the encoding cost increases as Subme does, except a little decrease when the level goes from 0 to 1. However, in Figure 12, there is a valley at level 5 on the bitrate curve, suggesting that the application of SATD and RDO in fact increases the bitrate. This is mainly due to the rate control method (CRF) of our choice, which focuses on perceptual quality rather than the objective quality PSNR, which these two techniques target to improve. Similarly, Figure 13 shows the average encoding cost under different quality levels.

*5.4. Trellis.* Trellis quantization is used to optimize residual DCT coefficients by reducing the size of some coefficients while recovering others to take their place. This technique can effectively find the optimal quantization for each macroblock to optimize the PSNR relative to bitrate. There are three options in x264: level 1 applies conventional uniform deadzone; level 2 uses trellis only on the final encode of macroblock, while level 3 does them all. In addition, trellis quantization in x264 requires CABAC (Context Adaptive Binary Arithmetic Coder), instead of the less-expensive and more-popular CAVLC (Context Adaptive Variable Length Coder). Thus, we introduce a level 0, which repeats level 1, only replacing CABAC with CAVLC.

Figures 14 and 15 show the encoding cost, rate, and bitrate with varying trellis levels. Trellis is similar to the case of Subme (Figures 11 and 12), in which the encoding cost grows as the level increases, but the bitrate fluctuates. Again, this suggests that the objective of trellis quantization (optimizing objective quality PSNR) does not always agree with the objective of the CRF method, which is the perceptual quality. This is further confirmed by our later observation on PSNR. We also see that CABAC does not cause additional speed loss while reducing bitrate by about 10%, which implies that CABAC should always be turned on. Not surprisingly, applying trellis quantization increases encoding time, especially for the level 3. In Figure 16, the encoding

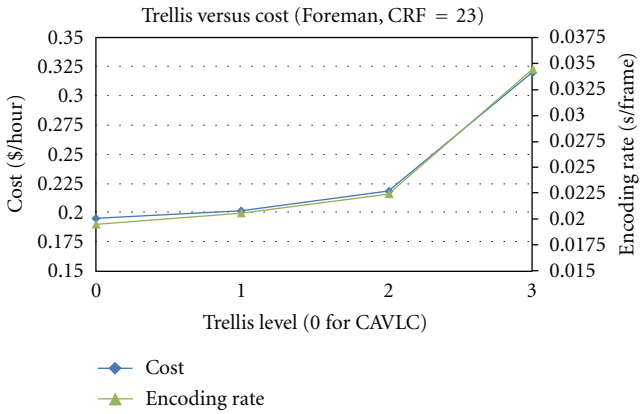



Figure 14: Encoding cost and rate with varying Trellis levels (Foreman, crf = 23).

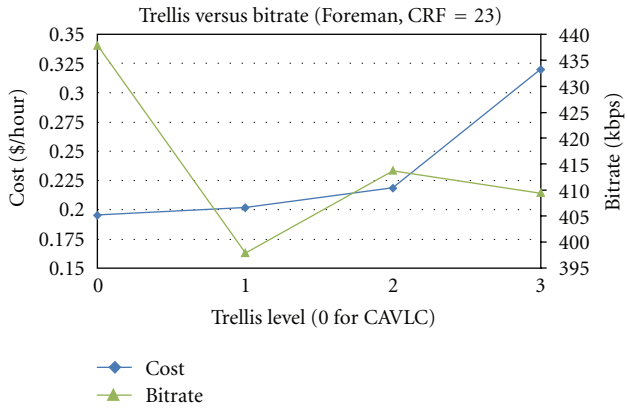



Figure 15: Bitrate and cost with varying Trellis levels (Foreman, crf = 23).

cost increases with the quality improvement, matching observations obtained from all previous experiments.

*5.5. CRF versus PSNR.* Although targeting at perceptual quality, CRF is known to be able to still achieve relatively constant objective quality level, that is, PSNR. To quantify the deviation, we observe PSNR in the above four parameter settings and obtain the following results in Table 1 (take Foreman, e.g.). The deviation is defined as the percentage of difference between lowest and highest PSNR to the highest value. We can see that during B-frame, Ref and Trellis tests, such change is within 1% range, while for Subme it is still around 3%. This makes us more confident to use CRF as the indication for quality level. Averaging all PSNR values from all tests gets us the correspondence between PSNR and CRF, as shown in Figure 17. The three quality levels show consistent rankings for all 12 videos, which implies that the encoding complexity varies with different videos.

*5.6. Composite Effect of Multiple Parameters.* In previous experiments, we only vary one parameter and leave all others default. Default setting uses 3 B-frames, 3 Refs, Subme level 7, and Trellis level 1. We plot the encoding costs from four parameters together in Figure 18. Approximately, the

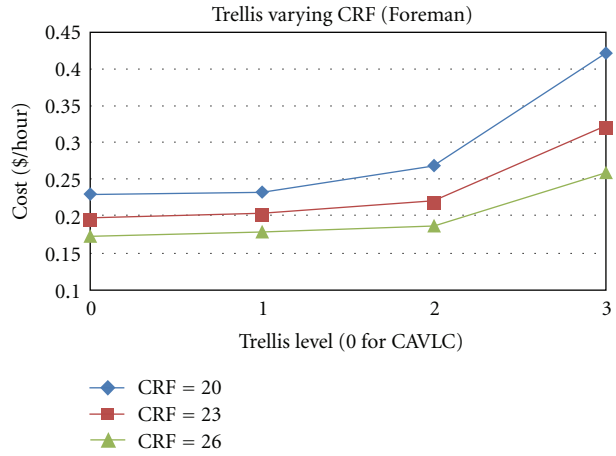



Figure 16: Encoding cost with varying Trellis levels under different CRF settings (Foreman).

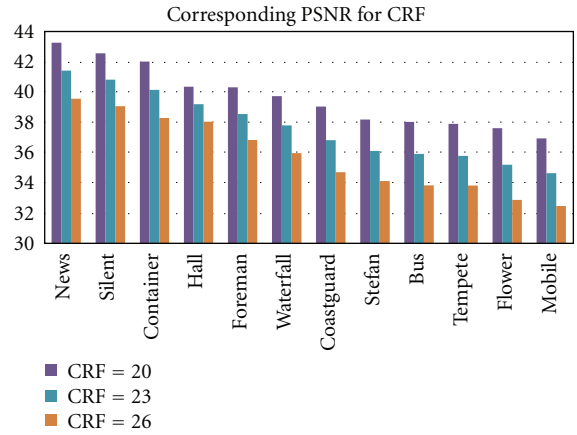



Figure 17: Correspondence between PSNR and CRF setting.

same costs are achieved with their respective default values. Subme leads to relatively lower costs while Ref renders linearly increasing costs with Ref number. The encoding costs incurred by all the settings, except for the cases of Trellis level 2, more than 3 Refs and Subme level 8 and 9, are bounded by the costs in B-frame tests. We also test for CRF value of 20 and 26, which show similar patterns. Therefore, we can define the baseline cost for Foreman under different quality levels as the averaged encoding cost in the B-frame tests. By this definition, our B-frame test (Figure 3) actually shows the baseline costs for all videos.

The encoding cost is influenced by the comprehensive impacts of multiple parameters. In order to understand how separate parameters collaborate with each other, in one experiment we vary all the four parameters, that is, B-frame from 1 to 3, Ref from 3 to 12, Subme level from 5 to 7, and Trellis level 0 and 1. There are 180 test points in total. The results are shown in Figure 19. We can see that the encoding cost shows a recurring pattern, which implies that the aggregated cost can probably be decomposed into separate weighted factors attributable to specific parameters.

Table 1: PSNR versus CRF (foreman).

| Foreman | crf = 20 | | crf = 23 | | crf = 26 | |
|---|---|---|---|---|---|---|
| | PSNR mean | Deviation (%) | PSNR mean | Deviation (%) | PSRN mean | Deviation (%) |
| B-frame | 40.38 | 0.51 | 38.60 | 0.09 | 36.88 | 0.33 |
| Ref | 40.45 | 0.79 | 38.73 | 1.08 | 37.03 | 1.21 |
| Subme | 40.05 | 2.6 | 38.25 | 3.01 | 36.49 | 3.26 |
| Trellis | 40.22 | 0.73 | 38.46 | 0.71 | 36.74 | 0.77 |

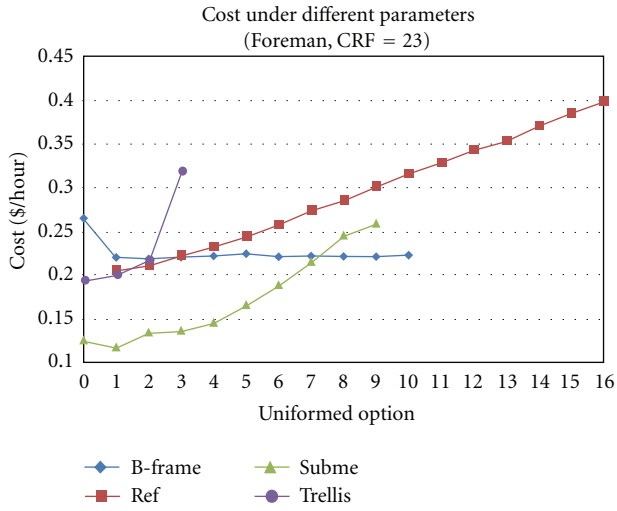



Figure 18: Encoding cost with four parameters setting together (Foreman, crf = 23).

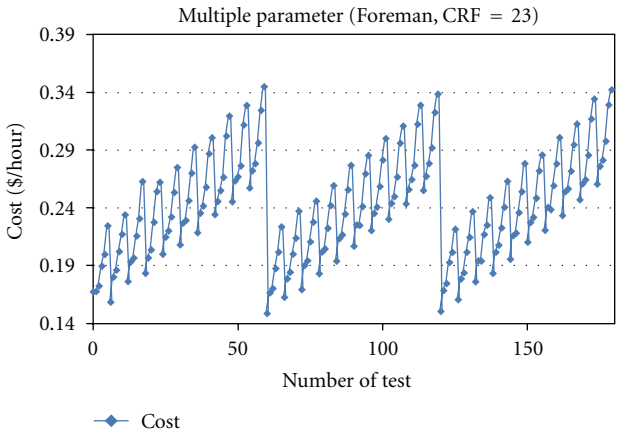



Figure 19: Encoding cost under combinations of four parameters (Foreman, crf = 23).

We also find that the average PSNR follows the similar pattern in Figure 20. PSNR ranges from 38.07 to 38.854, with the deviation of less than 2%. On a closer look, Trellis, Subme, and Ref consistently produce the expected impacts on PSNR.

*5.7. Other Parameters.* We have tested all major numerical parameters in x264, at least all that applicable to the CRF method. We here list other important parameters not included in our tests as below.

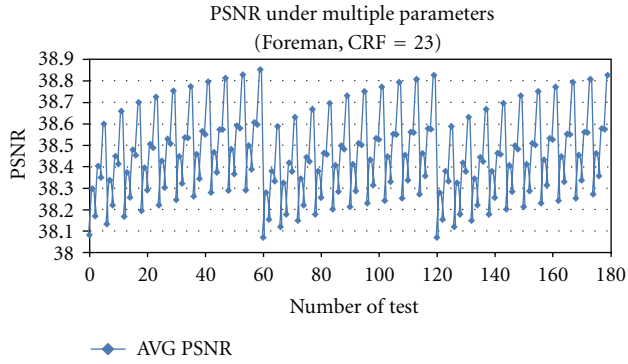



Figure 20: PSNR under combinations of four parameters (Foreman, crf = 23).

(i) ME determines the full-pixel motion estimation methods, and there are four choices in x264, namely, DIA (diamond), HEX (hexagon), UMH (Uneven Multi Hexagon), and ESA (Exhaustive Search). Generally, UMH provides good speed without sacrificing significant quality, while HEX is a better trade-off. We are using the default option HEX.

(ii) Partition controls the split up of 16 × 16 macroblock. Partitions are enabled per frame type and for intra- and interprediction, respectively. The default setting in x264 is "p8 × 8, b8 × 8, i8 × 8, i4 × 4." The p4 × 4 is not defaulted due to the high speed cost with respect to quality gain.

The above two parameters have also essential impacts on the encoding speed and quality. The reason why they do not fall into the scope of our consideration is that there is no apparent and absolute ranking among those options, which makes them hardly fit into our quantified parameter tuning framework. Besides, the empirical guidance about using the above two parameters are quite clear.

## 6. Discussion

Based on all the experiment results in the previous section, we claim the following observations.

(1) For all the tests conducted, if grouped by targeted CRF values, the resulting PSNR values in each group remain largely equal with a maximum deviation of 3% (Table 1).

(2) Each tested video, when trialed with different quality levels (CRFs), has its PSNR values shown with consistent ranking (Figure 17).

(3) Tests conducted regarding each parameter shows at least one period of asymptotic growth of encoding cost (e.g., when Ref ranges from 1 to 16), and one clear optimal point to minimize the cost (e.g., when Ref = 1). This observation also holds true across different quality levels tested.

The first two observations collectively confirm the effectiveness of CRF as a rate control method to accurately obtain consistent results on measurable and objective quality, that is, PSNR. Combined with the third observation, we can claim that our primary goal set at the end of Section 3 can be achieved. Conversely, with a clearly defined service model and desired quality level set by the user, our framework can find the optimal parameter setting to minimize the encoding cost.

However, this only works well in the best-effort way, that is, one can claim confidently that the encoding cost is indeed minimized, but never know how much the cost is until the job is done. This leads to the predictive power of the optimization framework, that is, can we predict the encoding cost of a video before it is encoded? This feature, if at all achievable, will be significantly valuable to the budget planning (hence projection of price and profit margin) for any service providers, especially appreciated when unclear service models are present, typically when it comes to the cloud-based distribution service whose storage and bandwidth costs are hard to determine. We further claim the following observations from our tests.

(4) By the functioning principles of block-based hybrid video coding, the tested parameters (e.g., Ref, Trellis, etc.) operate largely independent from each other. Our tests establish that, regarding each tested parameter, its asymptotic growth window (mentioned in claim (3)) is not affected by the changing of other parameters.

(5) Aside from independence, it is very likely that these parameters can be decomposed in a weighted sum fashion to qualify their contribution to the final encoding cost. Furthermore, such weights are not subject to the changing of the cost coefficients $\alpha$ and $\beta$.

(6) The B-frame test as the baseline cost (Figure 18) provides a worst-case envelope for the cost estimation if the prospect of claim (5) gets gloomy.

All these claims sound promising, even to a self-improving framework, which, with the growth of its sampling set, provides increasingly accurate weight estimation.

However, we must not ignore the elephant in the room. In Section 3, we mentioned the encoding complexity, which originates from the intrinsic characteristics of the video content itself. Evidently, Figure 21 shows the resulting bitrates of all 12 videos under the default setting (crf = 23) of x264, where the most complex video is encoded with

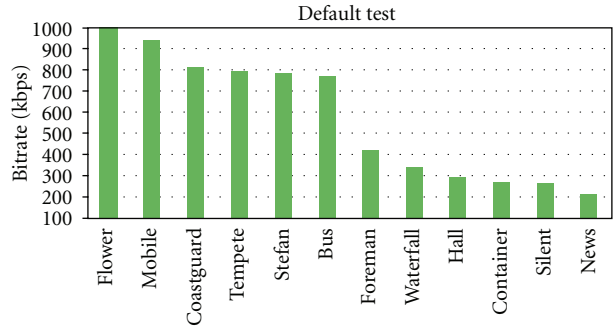



Figure 21: Resulting bitrates under the default setting of x264 for all videos.

bitrate 5 times the one of the lowest complexity. Figures 6 and 10 further confirm that this ranking holds across different quality levels, with the Hall as the only exception, indicating that the encoding complexity is largely rooted in the content itself.

There is no silver bullet to this problem. In order to learn the encoding complexity of a video, one has to encode it to know it, which paradoxically defeats the very propose of saving encoding cost. Fortunately, common practices of the modern encoding service largely alleviate this problem. First, most of the original content is already in the digital format (usually encoded by block-based hybrid video coding technology) awaiting transcoding to be suitable to play on the new platform. Second, the diversity of end-user terminals requires each video to be encoded multiple times to different qualities, resolutions, and formats. As such, we can easily obtain statistics from metadata of the originally encoded file to obtain the first-hand knowledge of the encoding complexity.

## 7. Conclusion

In this paper, we conduct an empirical study in the cloud-computing environment, targeting an optimization framework to save H.264 encoding cost by tuning the key parameters. Our study shows convincing results, in which the rate control method (CRF) demonstrates remarkable ability to precisely achieve the targeted quality level, and the tested parameters can be independently tuned to minimize the encoding cost. We have also discussed the encoding complexity as a major hurdle to the predictive power of our framework and how it can be alleviated by leveraging the diversity of end-user platforms.

Towards future study, the most urgent task is to quantify the contributing weight of key parameters to encoding cost, through more extensive experiments. Also we will incorporate important but nonnumerical parameters (such as motion estimation method) and examine its fitness to the current framework.

The final note is on the objective of our optimization framework, which is to minimize encoding cost, meanwhile maintaining certain quality level. It is not impossible for service providers to reverse the equation to give a cost budget,

try to maximize the encoded quality. In light of such a change, we should switch to rate control methods best at controlling bitrate, that is, ABR and VBR. Study of this flavor will be especially valuable when distribution cost becomes the overwhelming concern. Whether a similar parameter-tuning framework could be established on this regard could be only answered by another empirical study.

## References


[1] Amazon Web Services, http://aws.amazon.com/.

[2] Google Web Application Engine, http://code.google.com/appengine/.

[3] Microsoft Azure, http://www.microsoft.com/windowsazure/.

[4] http://www.youtube.com.

[5] http://www.youku.com.

[6] http://www.tudou.com.

[7] http://www.zencoder.com/flixcloud/.

[8] http://mewiki.project357.com/wiki/X264_Settings.

[9] A. Jagmohan and K. Ratakonda, "Time-efficient learning theoretic algorithms for H.264 mode selection," in *Proceedings of the International Conference on Image Processing*, pp. 749–752, October 2004.

[10] T. A. da Fonseca and R. L. de Queiroz, "Complexity-constrained H.264 HD video coding through mode ranking," in *Proceedings of the Picture Coding Symposium*, pp. 329–332, 2009.

[11] E. Akyol, D. Mukherjee, and Y. Liu, "Complexity control for real-time video coding," in *Proceedings of the IEEE International Conference on Image Processing*, pp. I77–I80, September 2007.

[12] X. Li, M. Wien, and J.-R. Ohm, "Rate-complexity-distortion optimization for hybrid video coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 21, no. 7, pp. 957–970, 2011.

[13] L. Su, Y. Lu, F. Wu, S. Li, and W. Gao, "Complexity-constrained H.264 video encoding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 19, no. 4, pp. 477–490, 2009.

[14] R. Vanam, E. A. Riskin, S. S. Hemami, and R. E. Ladner, "Distortion-complexity optimization of the H.264/MPEG-4 AVC encoder using the GBFOS algorithm," in *Proceedings of the Data Compression Conference*, pp. 303–312, March 2007.

[15] R. Vanam, E. A. Riskin, and R. E. Ladner, "H.264/MPEG-4 AVC encoder parameter selection algorithms for complexity distortion tradeoff," in *Proceedings of the Data Compression Conference*, pp. 372–381, March 2009.

[16] C.-E. Rhee, J.-S. Jung, and H.-J. Lee, "A real-time H.264/AVC encoder with complexity-aware time allocation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 20, no. 12, pp. 1848–1862, 2010.

[17] http://www.mplayerhq.hu.

[18] http://www.zencoder.com/.

[19] http://www.videolan.org/developers/x264.html.

[20] MSU Graphics & Media Lab (Video Group), "Fourth annual msu mpeg-4 avc/h.264 video codec comparison," http://compression.ru/video/codec comparison/mpeg-4 avc h264 2007 en.html.

[21] http://www.ffmpeg.org/.

[22] YUV Soft Inc., http://www.yuvsoft.com.

[23] L. Merritt and R. Vanam, "Improved rate control and motion estimation for H.264 encoder," in *Proceedings of the International Conference on Image Processing*, vol. 5, pp. 309–312, 2007.

[24] http://media.xiph.org/video/derf/.