Computers & Operations Research 40 (2013) 1289–1299

Contents lists available at SciVerse ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/caor

Optimum piece selection strategies for a peer-to-peer video streaming platform

Pablo Romero, Franco Robledo Amoza, Pablo Rodríguez-Bocca *

Laboratorio de Probabilidad y Estadística, Facultad de Ingeniería, Universidad de la República, Julio Herrera y Reissig 565, 11300 Montevideo, Uruguay

ARTICLE INFO

Available online 20 December 2012

Keywords:
Peer-to-peer
Piece selection strategies
COP

ABSTRACT

The client–server architecture is still popular due to its high predictable service and performance. However, it is not bandwidth scalable. An alternative setup for Internet video-streaming is offered by the peer-to-peer architecture, in which peers are servers as well as clients. Peers basically communicate in a three-level based policy. First, they meet other peers with common interests: this is called *swarming*. Then, each peer selects a small number of them for cooperation, called the *peer selection strategy*. In the last step peers cooperate sending pieces, defining the *piece selection strategy*.

This paper is focused on piece selection strategies. We propose an in-depth analysis of a simple cooperative model. In this model the issue is to find the best order in which pieces should be obtained. In the first stage, we introduce a Combinatorial Optimization Problem (COP), which maximizes the average user experience for video streaming services, and has a permutation as the decision variable. Its hardness motivates us to approximately solve it via an Ant Colony Optimization-based heuristic.

The main theoretical contributions are twofold: the introduction of a new piece selection strategy with better results in contrast with the ones found in the literature, and a systematic way of computing new piece selection strategies with high quality. The practical contribution is the incorporation of a new piece selection strategy in a live peer-to-peer streaming platform, with remarkable performance in relation with classical strategies.

1. Introduction

Internet-based multimedia systems have many different architectures, depending on their sizes and on the popularity of their contents. The majority of them have a traditional CDN (Content Delivery Network) structure [9,34], where a set of datacenters absorbs all the load, that is, concentrates the task of distributing the content to the customers. This is, for instance, the case of msnTV, YouTube, Jumptv, etc., all working with video content.

Another popular alternative consists in using the often idle capacity of the clients to share the video distribution with the servers through the present mature peer-to-peer (P2P) systems [24,16,18]. These are virtual networks developed at the application level over the Internet infrastructure. The nodes in the network, called peers, offer their resources to the other nodes, basically because they all share common interests. As a consequence, as the number of customers increases, the same happens with the global resources of the network. For this reason, P2P networks are said to *scale* well.

Nowadays P2P networks play an important role because of their popularity and their impact on Internet traffic. Some commercial P2P networks for live video distribution are available, all of them with proprietary source-codes and protocols. The most successful are PPlive [21,15], SopCast [28], PPstream [22], TVAnts [31] and TVUnetwork [32].

On one hand, the dynamism and freedom of peers in a P2P network are attractive and powerful tools for the users. On the other hand they impose many challenges on architecture design and protocols to share information [26,37]. The design of resilient peer-to-peer live streaming must cope with stringent timing constraints and node-churn, which is the unpredictable peer-arrival and departure. A recent survey on hints for a resilient design of streaming networks is [1]. A video frame has to reach its play-out time; otherwise the quality of experience is degraded. Moreover, many nodes only remain connected for a few minutes [29]. Several deployments and measures on P2P networks for live video distribution [27,30,2] confirm that the delay and play-out losses represent the most important factors in the quality of experience perceived by end users (see also [25,19] for related details). Therefore, a well-designed piece selection strategy is essential in order to obtain high play-out continuity and low latency in a P2P streaming network [8].

* Corresponding author. Tel.: +598 2 7114244.
E-mail address: prbocca@fing.edu.uy (P. Rodríguez-Bocca).

0305-0548/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cor.2012.12.008

There are three kinds of streaming services, that differ in generation, distribution and synchronization between peers, to know: file sharing, video on-demand and live-streaming. In file sharing, the file is available only after complete downloading. In video on-demand, the stream is distributed only when users demand it. The third service is live-streaming. Here, all users must be synchronized watching at the same instant, and the video stream is distributed and generated simultaneously. An inspirational system for file sharing and fast propagation is called BitTorrent [8]. There are many papers that show BitTorrent works for both off-line and on-demand services. However, it is not well-adapted for live-streaming requirements. An enormous effort of the scientific community is pointing to understand BitTorrent's deficiencies, and finally provide a full-scalable triple-play BitTorrent compatible with the three streaming modes. Diverse mathematical models try to understand the behavior and scalability of BitTorrent-based systems, including Markov Chains [38], Fluid Models [23], Branching Processes [33,36] and Marginal Probabilities [39], among many others.

We developed an in-depth analysis of the model stated in [39] mainly because it is simple and captures two fundamental notions of live-streaming scalable architectures: cooperation and synchronization. There, the authors designed a cooperative pull process, where peers cooperate with each other in order to recover a video streaming delivered by a single source-node. The time is discretized, and requesting peers use a random peer selection policy to request for a new video piece. The aim is to find an optimal piece selection strategy, that dictates the order in which pieces must be downloaded to achieve high continuity and low buffering times. In this paper, we present a new strategy that has better results than previously considered proposals. For its design, we define a Combinatorial Optimization Problem (COP), translated into a suitable formulation of an Asymmetric Traveling Salesman Problem (ATSP). The latter is solved meta-heuristically following an Ant Colony Optimization (ACO) approach, which is inspired in the way ants find the shortest path between their nests and their food [4]. We refer the reader to [10–12,7] for an in-depth analysis of this nature-inspired heuristic.

This paper is organized as follows. Section 2 describes a simple model of piece selection strategies proposed in [39]. Section 3 introduces a new family of piece selection strategies, its basic properties and an ideal approach. Section 4 contains a Combinatorial Optimization Problem (COP) whose decision variables are permutations. A methodology for its meta-heuristic resolution is also developed here. This process enables to obtain new piece selection strategies. Section 5 contrasts the new piece selection strategy with classical ones from both theoretical and empirical aspects. In the light of the mathematical model, we present analytical comparisons between the proposed strategies and previous ones. Then, the new piece selection strategy is applied into a real peer-to-peer platform called GoalBit [14,6], showing the practical advantages of the new strategy proposed. Finally, Section 6 contains the main conclusions of this work.

2. Mathematical model

The description of this mathematical model is taken from [39] and its most modern version, detailed in [38]. Consider a closed fully connected P2P live-streaming system which needs to serve M identical peers. This P2P system has a logical server $\mathcal{S}$, which organizes the video content into a stream of pieces, sent in playback order. We model this large scale network as a discrete-time system. At each time slot, the server $\mathcal{S}$ uploads one video piece to one peer uniformly chosen at random. Each piece has a sequence number, starting from 1. Therefore, at time slot t, the server randomly selects one peer and uploads the video piece of sequence number t to this randomly selected peer.

Each peer needs to receive and buffer these video pieces from the P2P streaming system. To achieve this, each peer holds a local buffer $\mathcal{B}$, which can cache up to N video pieces. Position $\mathcal{B}_1$ stores the newest video piece that the server $\mathcal{S}$ is uploading in the current time slot, whereas position $\mathcal{B}_N$ is used to store the oldest video piece, that is currently being played back. In other words, when server $\mathcal{S}$ is uploading a piece with sequence number $k \geq N$, the video piece $k-N+1$ is being played back by the peer (provided that the video piece is available in $\mathcal{B}_N$). At the end of each time slot, the video piece in $\mathcal{B}_N$ is discarded, and all pieces will be "shifted right" by one buffer position: video piece in $\mathcal{B}_i$ will be shifted to $\mathcal{B}_{i+1}$ for each $i=1,\ldots,N-1$. A distortion is perceived on the screen if the peer could not obtain the current piece in time. We assume that all peers are synchronized in the buffer consumption. See Fig. 1 for a graphical description.

Peers need to collaborate with each other to minimize their chance of losses and delays. As a consequence, pieces can also be obtained from other peers in a pull-based process (i.e. pressed by downloader needs). Peer A randomly selects another peer B within the network. In general, and following some specific strategy, peer A will look at a given position in its buffer. If it is empty, it will request peer B for this missing piece. If the initial buffer position is already filled, the requesting peer shifts to some other position and the same iteration is repeated until it finds an empty one. If B does not have the corresponding piece, then A can ask for another piece. This scheme leads either to a success, when A finally gets a video piece from B, or to a failure, when there is no empty position in A's buffer that can be filled by a piece coming from B. The *extension* of the query is the number of buffer positions that A needs to examine in order to get a new piece. It is assumed that the whole query lasts no more than one time slot, and every peer obtains no more than one piece during a time slot (the peer chosen by the server does not ask for more video pieces).

Let us call p_i the probability that a peer has the correct video piece in $\mathcal{B}_i$. By symmetry, if all peers use the same strategy, p_i is independent of the peer. In stationary state, it does not depend on time either. Consider that a particular peer A selected a peer B to download a piece. Using a specific piece selection strategy, suppose that at some time, the piece corresponding to A's buffer position $\mathcal{B}_i$ is missing, so, it is desired by peer A and owned by peer B. The probability of this event is denoted by s_i (for the reasons mentioned above, it only depends on i).

Definition 2.1. The *buffer-map* $(p_1,\ldots,p_N)$ is the probability that peer owns a video chunk in the buffer-cell $\mathcal{B}_i$. The number $C=p_N$ is the playback-delivery ratio, or *continuity*.

In symmetric conditions, peers will have the same buffer-map p_i, and the buffering time can be measured [39].

Definition 2.2. The *buffering-time* or latency for a peer can be found by $L=\sum_{i=1}^{N} p_i$, and represents the expected time (measured in slots) a joining peer in the system should wait in order to reach the buffer-map p_i, starting with an empty buffer.

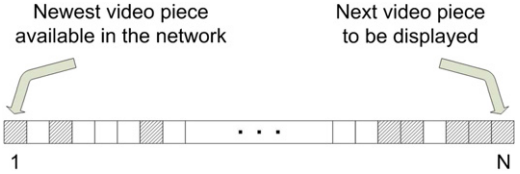

Fig. 1. Buffer model for each peer. Position $\mathcal{B}_1$ has the newest video piece in the buffer, and $\mathcal{B}_N$ the piece being displayed at the screen.