STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LIX**, Number 2, 2014

# CHOOSING THE BEST SET OF ACTIVE SOURCES IN PEER-TO-PEER VIDEO STREAMING

CLAUDIU COBÂRZAN, ADRIAN STERCA, AND CRISTINA MIHĂILĂ

Abstract. We present some of the challenges of streaming video data in Peer-to-Peer systems, more specifically choosing the optimal set of active streaming sources for a video. This problem is modeled like a *knapsack-like problem* and solved using a genetic algorithm. We discuss the modeling process with emphasis on benefit and cost, the new fitness function which guides the deployed genetic algorithm as well as the obtained results with generated test data.

## 1. Introduction

Peer-to-Peer (P2P) networks (e.g. [11]) are popular with data distribution since they offer scalability, robustness and low cost when compared with traditional data distribution architectures, most of which consider the client-server model. As a result of increasing demand for multimedia data in the Internet, numerous approaches to audio-video streaming have been proposed in the past decade. Among those, of growing popularity is the use of P2P streaming.

In a P2P video streaming system, each peer requesting the streaming of a video can get the video data from $n$ sources which locally store either the whole video stream or parts of it. Chunks of video data can be streamed in parallel from those $n$ peer sources and played according to the playout time by the local video player.

The requesting peer has the option to download video data (segments) from a single peer, multiple peers from those $n$ sources holding the video stream or all $n$ sources. It is important to note that choosing to download video data from as many sources as possible is not necessarily optimal streaming. Each source that streams video data to be played by the requesting peer in

Received by the editors: September 30, 2014.

2010 *Mathematics Subject Classification.* 68M14, 68M20.

1998 *CR Categories and Descriptors.* C.2.4 [**Computer Systems Organization**]: Computer-Communication Networks – *Distributed Systems*; C.2.3 [**Computer Systems Organization**]: Computer-Communication Networks – *Network Operations*.

*Key words and phrases.* peer-to-peer networks, video streaming.

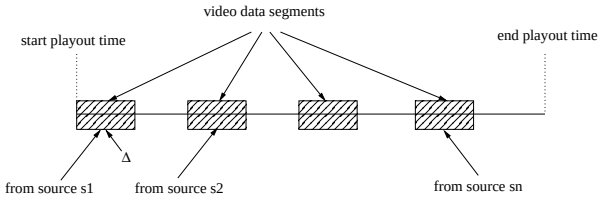


FIGURE 1. Buffering video segments from multiple P2P sources.

the distant future consumes bandwidth which otherwise would be available to the source peers that stream video data for playout in the near future.

More explicitly said, adding a new streaming source to the set of active streaming sources has the benefit of a reduced delivery time of video data since there are more sources which stream video data in parallel. The cost of adding a new source to the set of active streaming sources is reflected by the fact that this new source consumes local bandwidth which otherwise would be available to the other streaming sources (i.e. it reduces the effective bandwidth of the other streaming sources including the one that streams the data which will be first played out).

The rest of the paper is structured as follows: in Section 2 we review related work, then in Section 3 we model the problem of choosing the optimal number of sources to stream from, while Section 4 presents the genetic algorithm we have deployed for solving the problem. Experimental results are presented and discussed in Section 5. Section 6 concludes the paper and hints on future work.

## 2. Related work

In spite of their obvious advantages, P2P systems have their own problems like high churn rates, heterogeneous peers and frequent node failures which are largely due to peer selfishness (once a streaming session concludes, or a download finishes, the peer disconnects). There exist approaches which try to address those problems by employing scalable video coding and network coding within P2P systems [9] or by offering hybrid solutions which include the use of proxy-clients in conjunction with P2P systems [6] as well as different caching strategies (suffix caching [2], demand based caching [7], [10], [5]).

In live P2P streaming there are two approaches for selecting the neighboring peers which deliver a specific video to a requesting node: *tree-based approaches* and *mesh-based approaches* [8]. In tree-based P2P streaming data travels from a source provider to the receiving peers in a tree-like overlay network in which video data is pushed from the source (i.e. the tree root) down the tree. The mesh-based streaming approach organizes peers sharing the

same content in randomly connected meshes and employs a swarming content delivery mechanism over each such mesh. For Video-on-Demand (VOD) P2P streaming the mesh-based approach is more frequently used. However, to the best of our knowledge, the number of peers and the peers themselves that form such a mesh are chosen randomly, the only criteria being that they share the same content and their locality [1].

## 3. Problem modeling

In this paper we try to determine an optimal number of peers and the peers which should form the source mesh for a VOD P2P streaming. This problem must be solved by a node from the P2P system wanting to watch a video which has to decide how many and which sources to stream this specific video from in the P2P system. We model the problem as a *knapsack-like problem*. The objects which are considered here are the peers owning a specific, generic, video stream in a P2P network. We associate to each of these source peers a benefit value (given as the value of a function) for adding it to the set of active streaming sources, but also a cost value (also given as the value of a function). The effective utility of a source peer for the peer wanting to watch the video will be given by its benefit/cost ratio.

We denote by $S$ the set of streaming sources (peers in the P2P system), $|S| = n$, that store the video stream requested by a specific peer $p$. In order to make the model as general as possible, we do not impose any restrictions on the codecs used for the video streams.

Each peer $s$ in the P2P system is characterized by the following characteristics:

- *bandwidth* - the amount of data than can be streamed (measured in bytes/second) from $s$ to the requesting peer $p$; initially, the exact bandwidth can not be known by $p$ but it can be estimated after several streaming rounds; initially, each peer advertises its maximum upload bandwidth (as provided by its ISP) although this is not the real bandwidth the receiving peer can use, since it depends on network conditions (e.g. congestion, shared network links etc.) which lower this maximum value;
- *rtt (round-trip time)* - this is advertised by $s$ initially;
- *jitter* - the variation of delay in the network path from $s$ to $p$; it is computed by $p$.

The receiving peer $p$ has a buffer which holds the video data from various segments received so far from the sources in set $S$. This buffer is drained as the data is played by the local video player. The process of buffering segments from multiple sources is depicted in Figure 1.

We denote by $\Delta$ the size (in seconds of video data) of the video segment in the buffer which will be played first.

When a new source $s_k$, $1 < k < n$ is added to the set of active streaming sources, the benefit for the requesting peer $p$ is given by the following function:

$$U(s_k) = \frac{a \cdot bandwidth(s_k)}{b \cdot rtt(s_k) + c \cdot jitter(s_k)} \tag{1}$$

where $U : S \to (0, \infty)$ and:

- $a$, $b$, $c$ are positive constant weights normalized to $[0, 1]$;
- $bandwidth(s_k)$ is the bandwidth of the peer $s_k$ with respect to the requesting peer $p$;
- $rtt(s_k)$ is the round-trip time of the network route between $s_k$ and $p$;
- $jitter(s_k)$ represents the variation of $rtt(s_k)$ on the route between $s_k$ and $p$.

Intuitively, the benefit of adding a peer $s_k$ to the set of active streaming sources is directly proportional to its bandwidth with respect to the requesting peer (i.e. $bandwidth(s_k)$), but also inverse proportionally to its round-trip time and jitter because a source with a high round-trip time and jitter is less reliable in rapidly adapting to changes in network conditions (i.e. level of congestion, available bandwidth, link failure etc.). The wighting constants, $a, b$ and $c$, are included in the formula because they add to the generality of the model: one requesting peer might value more the available bandwidth for receiving a video from a source peer than its round-trip time, other requesting peer might do the opposite.

Conversely, the cost of including the same source $s_k$ in the set of active streaming sources depends on the amount of bandwidth it "steals" from the already included sources, the number $k$ of streaming sources in the set $S$ and the buffer size $\Delta$ and is expressed as:

$$C(s_k) = \frac{1}{\Delta} \cdot k \cdot \sqrt{(total\_bandwidth - max\_download\_bandwidth)^+} \tag{2}$$

Intuitively, the more (receiving) bandwidth of peer $p$ is used by the source $s_k$, the larger is the cost for adding source $s_k$ to the set of active streaming sources.

where $C : S \to [0, \infty)$ and:

- $total\_bandwidth = \sum_{i=0}^{k} bandwidth(s_k)$;

- *max_download_bandwidth* is either the downloaded bandwidth provided by the ISP (initially) or the maximum download bandwidth reached during the streaming session so far.

The upper + sign has the following meaning $(X)^+ = max(0, X)$.

The cost function is increasing with $k$, the number of sources in $S$ - the set of active streaming sources, the total bandwidth of the streaming sources in $S$ and inversely proportional with $\Delta$. Note that the cost is 0 when the $total\_bandwidth < max\_download\_bandwidth$. This happens when there are few streaming sources in $S$.

With the above notations we model the problem of choosing the cardinal $k$ and the set of active streaming sources $S$ ($|S| = k$) from all $n$ peers that have the requested video stream, as a *knapsack-like problem*. The idea is to continuously add streaming sources to the set $S$ (similar to adding various objects to the knapsack) as long as the following condition is satisfied:

$$\sum_{i=0}^{k} C(s_k) \leqslant max\_download\_bandwidth \tag{3}$$

The goal is to achieve the maximum benefit while respecting the constraint in (3).

We solve this problem by using a genetic algorithm [3], [4] which is presented in the next section. The algorithm will be applied periodically, in rounds, during the peer-to-peer streaming session at a coarse time granularity (e.g. from 10 to 10 minutes). Of course, the execution time of the algorithm must be lower than the time it takes the Peer-2-Peer network to change its content (i.e. when a peer exits the network or removes a shared file).

## 4. The algorithm for choosing the set of active sources

The set of active streaming sources $(s_1, s_2, ..., s_k)$ and its cardinal $k$ will be determined using a genetic algorithm. The algorithm starts with a population of randomly generated solutions (chromosomes). The solution is represented as a binary string (chromosome) of length equal to $n$ - the number of peers that can serve the requested video stream. The value corresponding to each position $i$ in this binary string indicates if the $i^{th}$ peer is in the selection or not (1 or 0).

A new population is generated by applying the following genetic operators [3], [4]: binary tournament selection for parent selection, and one point crossover and strong mutation for generating new offspring chromosomes that will form a new population. The evolution process is similar to the evolution

scheme of a standard genetic algorithm and additionally an elitism selection is used.

The formula used to compute the fitness of each chromosome is as follows:

$$fitness(S) = \begin{cases} f(S), \\ \qquad \text{if f(S)} \leq \text{max\_download\_bandwidth} \\ max\_download\_bandwidth - f(S), \\ \qquad \text{if f(S)} > \text{max\_download\_bandwidth} \end{cases} \tag{4}$$

where

$$f(S) = \sum_{i=0}^{k} C(s_k), \forall s_k \in S \tag{5}$$

and $S$ is a possible solution represented by a chromosome and $C(s_k)$ is described in (2).

The quality of a solution (chromosome) is its total utility. The goal is to maximize this value while satisfying the constraints imposed by (3).

## 5. Experiments and evaluation

Several tests were conducted in order to evaluate the proposed method. We have used the following values for the fixed parameters in the benefit and cost functions:

- : $a = b = 0.4$, $c = 0.2$;
- : $\Delta = 10$, in seconds;
- : $n = 1000$;
- : $max\_download\_bandwidth = 3000$, in kilobits/second.

The choice of $a$, $b$ and $c$ favors the *bandwidth* and the *rtt* in the utility function and gives only a small weight to the *jitter*. This is because *jitter* can usually be compensated using buffers at the video player. The value for the *max_download_bandwidth* is typical for a medium-cost network connection and is scaled with the *bandwidth* values for the peers, $s_k$. The value of 10 seconds for $\Delta$ is a typical minimum buffer value considered relatively safe before the client starts playing the video.

For each node $s_i$, $1 \leqslant i \leqslant n$ we have generated random values with the following characteristics:

- : $bandwidth(s_i) \in [50, 400]$, in kilobits/second;
- : $rtt(s_i) \in [0.02, 1]$, in seconds;
- : $jitter(s_i) \in [0.2, 0.5]$, in seconds.

The random generated values are between limits considered typical for a metropolitan area network.

Each experiment was repeated 10 times with different random seeds and each run was conducted with the parameter settings specified in Table 1.

TABLE 1. Parameter settings

| Parameter | Value | | | |
|---|---|---|---|---|
| Number of generations | 250 | 500 | 750 | 1000 |
| Population size | 15 | 15 | 15 | 15 |
| Crossover probability | 0.5 | 0.5 | 0.5 | 0.5 |
| Mutation probability | 0.002 | 0.002 | 0.002 | 0.002 |

In the evolutionary algorithm we used formula 4 for computing the fitness of each solution.

The best and average results and the standard deviations obtained for each test instance are displayed in Table 2 and graphically in Figure 2 and Figure 3.

TABLE 2. Obtained results for the settings in Table 1

| Number of generations | Best utility | Average utility | Standard deviation | Time (seconds) |
|---|---|---|---|---|
| 250 | $-13,912.20$ | $-19,304.30$ | $3,113.04$ | 175 |
| 500 | $16,172.30$ | $4,519.29$ | $7,270.34$ | 341 |
| 750 | $17,093.00$ | $15,250.50$ | $1,624.40$ | 515 |
| 1000 | $18,387.20$ | $17,524.10$ | 652.32 | 695 |

## 6. Conclusion and Future Work

We have presented an algorithm and a model for the problem of choosing the best set of streaming sources for a video in a Peer-to-Peer system. The problem was modeled as a *knapsack-like problem* and we devised a genetic algorithm to solve this problem. The model proposed is a better alternative to the random choice of source peers when forming the set of active streaming sources for a requested video stream, because it optimizes the utility of the chosen streaming sources set for the receiving node.

As future work we intend to implement the proposed algorithm in a P2P sharing system like Bittorent [11] and test it in real network conditions.

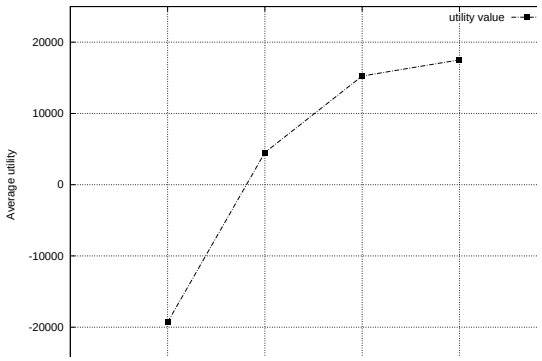


FIGURE 2. Average utility value variation with the number of generations

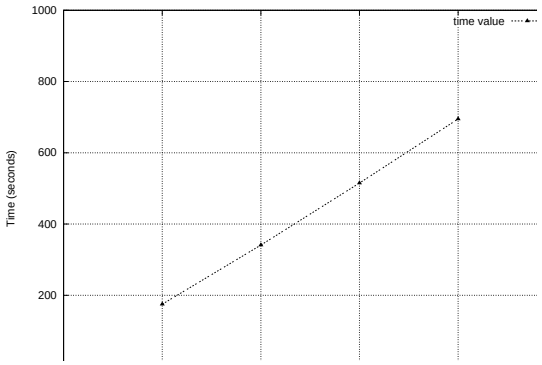


FIGURE 3. Time variation (until the experiment is completed) with the number of generations

## ACKNOWLEDGMENTS

This work was partially supported by the CNCSIS-UEFISCSU unit of the Romanian Government, through project PN II-RU 444/2010.

## REFERENCES

[1] M. Alhaisoni, M. Ghanbari, A. Liotta (2009). *Streaming Layered Video over P2P Networks*. In Proceedings of the 7th International Conference on Advances in Mobile Computing and Multimedia, pp 76–81.

[2] H. Alustwani, J. M. Bahi, and A. Moustefaoui (2008). *Improving data availability in p2p streaming systems using distributed virtual cache.* In Tenth IEEE International Symposium on Multimedia, pp 384–389.

[3] Thomas Baeck, David B. Fogel, and Zbigniew Michalewicz, editors. *Evolutionary Computation 1, Basic Algorithms and Operators*. Institute of Physics Publishing, Dirac House, Temple Back, Bristol BS1 6BE, United Kingdom.
[4] Thomas Baeck, David B. Fogel, and Zbigniew Michalewicz, editors. (2000) *Evolutionary Computation 2, Advanced Algorithms and Operators*. Institute of Physics Publishing, Dirac House, Temple Back, Bristol BS1 6BE, United Kingdom.
[5] Y. He and Y. Liu. Vovo (2009). *Vcr-oriented video-on-demand in large scale peer-to-peer networks.* In IEEE Transactions on Parallel and Distributed Systems, pp 528–539.
[6] A. T. S. Ip, J. Liu, and J. C.-S. Lui. (2007). *COPACC: An architecture of cooperative proxy-client caching system for on-demand media streaming.* In IEEE Transactions on Parallel and Distributed Systems, pp 70–83.
[7] U. C. Kozat, O. Harmanci, S. Kunumuri, M. U. Demircin, and M. R. Civanlar (2009). *Peer assisted video streaming with supply-demand-based cache optimization.* In IEEE Transactions on Multimedia, pp 494–508.
[8] N. Magharei, R. Rejaie and Y.Guo (2007). *Mesh or Multiple-Tree: A Comparative Study of Live P2P Streaming Approaches.* In IEEE INFOCOM 2007, 26th IEEE International Conference on Computer Communications, pp 1424–1432.
[9] S. Mirshokraie and M. Hefeeda (2010). *Live Peer-to-Peer Streaming with Scalable Video Coding and Networking Coding.* In MMSys'10 Proceedings of the first annual ACM SIGMM conference on Multimedia systems, pp 123–132.
[10] H.-S. Tang, S.-H. G. Chan, and H. Li (2009). *Optimizing segment caching for peer-to-peer on-demand streaming.* In IEEE International Conference on Multimedia and Expo, pp 810–813.
[11] The Bittorent Protocol Specification, `https://wiki.theory.org/BitTorrentSpecification`

Babeş-Bolyai Univesity, Faculty of Mathematics and Computer Science, Department of Computer Science, 1 M. Kogălniceanu St., 400084 Cluj-Napoca, Romania

*E-mail address*: `claudiu@cs.ubbcluj.ro`
*E-mail address*: `forest@cs.ubbcluj.ro`
*E-mail address*: `cristina.mihaila@cs.ubbcluj.ro`