A GAME THEORETIC FRAMEWORK FOR OPTIMAL RESOURCE ALLOCATION IN P2P SCALABLE VIDEO STREAMING

Stefano Asioli, Naeem Ramzan, and Ebroul Izquierdo

School of EECS, Queen Mary University of London, London, UK
{stefano.asioli, naeem.ramzan, ebroul.izquierdo}@eecs.qmul.ac.uk

ABSTRACT

In this paper we describe a game theoretic framework for scalable video streaming over a peer-to-peer network. The proposed system integrates optimal resource allocation functionalities with an incentive provision mechanism for data sharing. First of all, we introduce an algorithm for packet scheduling that allows users to download a specific sub-set of the original scalable bit-stream, depending on the current network conditions. Furthermore, we present an algorithm that aims both at identifying free-riders and minimising transmission delay. Uncooperative peers are cut out of this system, while users upload more data to those which have less to share, in order to fully exploit the resources of all the peers. Experimental evaluation shows that this model can effectively cope with free-riders and minimise transmission delay for scalable video streaming.

Index Terms— peer-to-peer, scalable video, game theory

1. INTRODUCTION

In the past few years, popularity of multimedia applications over the Internet has grown exponentially. This is due to the increasing diffusion of broadband connections. Highly demanding services for video on demand or live streaming are now offered by several providers and most of them usually rely on a client/server architecture. However, this approach is expensive and it does not exploit the idle resources of the users of the system. On the other hand, peer-to-peer (P2P) is a valid alternative to this model. In fact, if peers are given the proper incentives, they can share their resources, getting a reward from other users and lowering the burden of the server.

Live video streaming over P2P networks, however, still presents some challenges. First of all, data chunks have strict deadlines, represented by their playback time. That is, in real-time applications like P2P TV broadcasting, the delay between the generation of a chunk and the receiving time should be minimised. Second, users have different upload capacities and P2P networks are highly heterogeneous. Finally, users in this system can behave as free-riders. In fact, knowing the vulnerabilities of a P2P protocol, they download data without returning anything in exchange.

The problem of network heterogeneity can be partially solved by using scalable video coding (SVC). This allows to adapt the content to different users requirements by selecting an appropriate subset of the original sequence for download and discarding the rest. These codecs are based on the DCT transform, like the standard H.264/SVC [1], or wavelet-based [2]. In both cases, adaptation can be performed in terms of frame size, frame rate or Signal-to-Noise Ratio (SNR).

This research was partially supported by the European Commission under contract FP7-248474 SARACEN and FP7-287723 REVERIE.

P2P systems for video streaming can be divided into two categories: tree-based and mesh-based. A few algorithms for minimum-delay streaming rely on a tree-based architecture [3]. This approach has several advantages, including better resource allocation. However, especially in big networks, these trees are expensive to maintain. Furthermore, the most critical aspect is the lack of resource reciprocation between a child and a parent node. A solution to this problem is represented by complementary trees [3]. However, this system needs to create several overlay networks that need to be constantly balanced. On the other hand, there exist solutions based on a mesh topology. Specifically, [4] is a *push-based* solution where the latest useful chunks are forwarded to the *most deprived* peers. Experimental evaluation shows that this technique can achieve results that are comparable to tree-based approaches.

The problem of resource reciprocation in P2P networks has been an important area of research for many years. Several models, like the original BitTorrent (BT) [5] focus on short-term effects, which might not be suitable for video related applications. In fact, peers might not always have data to share with users they are currently downloading from. Many alternative approaches are based on game theory. In [6] a credit-line mechanism is introduced. In this solution, peers always cooperate with other users, unless the contribution of a user becomes too big with respect to its reciprocation. This strategy is cheat-proof, a Nash Equilibrium and strongly Pareto optimal [6].

The main contribution of this paper is a game theoretic framework for scalable P2P video streaming with a focus on optimal resource allocation and delay minimisation. To the best of our knowledge, optimising streaming rates and finding incentives for users to cooperate are usually considered as separate problems. Most real systems, however, need to deal with both at the same time. Our technique considers both issues, while aiming at achieving nearly optimal performance. In addition to this, our approach is designed for the transmission of scalable video sequences, as compared to conventional video streaming services. In our proposed architecture, peers are discouraged from downloading videos with a bit-rate which is higher than their upload capacity. The result is a system in which misbehaving users are detected and cut out, while the others are rewarded by achieving real time streaming.

The remaining Sections of this paper are organised as follows: Section 2 provides some background; Section 3 describes the proposed algorithm; Section 4 presents the results obtained in our simulation; Finally, Section 5 concludes the paper.

2. BACKGROUND

The problem of finding incentives in P2P networks has been extensively studied in the past few years. More recently, some systems have been specifically designed for video streaming. A common fac-

tor of all these solutions is that each peer is associated with a *utility* or *payoff function* [7]. Let us suppose that a video bit-stream of bit-rate q is divided into chunks and this video is originally stored in a server; for simplicity, time is divided into rounds. For the two-player game, the utility function for a single round is defined as follows [7]:

$$u_1(x_1, x_2) = x_2\ \gamma_1 - x_1\ \lambda_1$$
$$u_2(x_1, x_2) = x_1\ \gamma_2 - x_2\ \lambda_2 \quad (1)$$

In (1), x_i is the action taken by peer P_i and it is either 1 or 0, depending on the decision of i to cooperate or defect. $\gamma \in [0, 1]$ denotes the gain which a certain user obtains from the receiving a video chunk. On the other hand, λ_i is the loss associated to cooperation.

P2P video streaming can be considered as an infinitely repeated game [6]. In this case, an averaged utility function $U_i(\mathbf{x})$ is considered, where $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2)$ is a vector containing the decisions of the peers at each round. This function is defined as:

$$U_i(\mathbf{x}) = \frac{\sum_{t=1}^{T_{fin}} u_i(\mathbf{x}_1(t), \mathbf{x}_2(t))}{T_{fin}} \quad (2)$$

where $\mathbf{x}_i(t)$ is the action of peer P_i at round t and T_{fin} can be either the round corresponding to the end of the playback or the round in which peers stop cooperating. In [6] this happens when the other peer refused to cooperate in the previous round or if the other user cannot share any data that can be useful for the current peer. If both peers cooperate until the end of the game, the corresponding utility function will be $U(\mathbf{x}) = (U_1(\mathbf{x}), U_2(\mathbf{x})) = (\gamma_1 - \lambda_1, \gamma_2 - \lambda_2)$.

In the multiple user scenario, however, a cooperating peer might not always be able to reciprocate resources immediately. Therefore, this *one strike and you are out*-like strategy might not be optimal. A credit line mechanism is therefore proposed and a maximum debit $\Delta^{\max}$ is introduced. In this case, considering a peer P_i and the set of its neighbours $P_k \in N_i$, where N_i is the set of neighbour peers of P_i, the number of chunks P_i uploaded to a generic neighbour P_k can be written as $\chi_k^s(i,t)$, while the number of chunks received by the same peer is denoted by $\chi_k^r(i,t)$. In order to identify free-riders in the system, P_i adopt the following rule:

- if $\chi_k^s(i,t) - \chi_k^r(i,t) \leq \Delta^{\max}$ then P_i marks P_k as cooperative.
- if $\chi_k^s(i,t) - \chi_k^r(i,t) > \Delta^{\max}$ then P_i marks P_k as free-rider.

This implies that the number of chunks two peers exchange should be asymptotically equal. It is possible to prove [6] that in case of no attacks this strategy is sub-game perfect, cheat-proof and, if none of the peers stops receiving chunks, strongly Pareto optimal.

3. PROPOSED MODEL

The proposed system consists of several parts. First of all, the architecture can be classified as pull-based; the video source uploads the content to selected peers, which subsequently receive requests from their neighbours. Our proposed algorithms concern packet scheduling and resource allocation. Content adaptation is performed by using a wavelet-based layered scalable video codec [2], while our scheduling technique is based on a sliding window [8]. This allows the users to give different priorities to different sub-sets of the bit-stream, in order to maximise the received video bit-rate with respect to the available resources. Moreover, as far as upload bandwidth allocation is concerned, minimum-delay streaming is achieved uploading data to the most deprived peers [4]. In fact, uploading data to peers that have less to share increases the chances of them having something to upload in the future [4]. Finally, free-riders are cut out using a credit line mechanism [6]. In our system, however, peers are allowed to send low or high priority requests in order to fully utilise the capacity of the network.

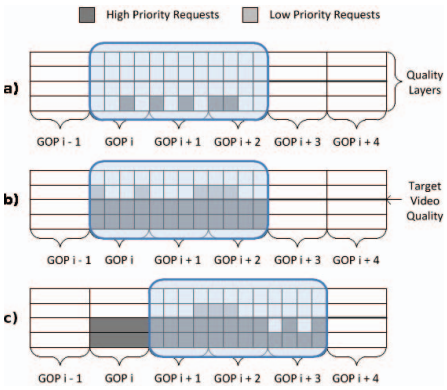

Fig. 1. Sliding window for scalable live video streaming, showing high and low priority requests.

3.1. Packet Scheduling Algorithm

The proposed solution has been designed for the Wavelet-based Scalable Video Codec (WSVC) developed at our institution [2]. However, a similar solution can be implemented for the standard H.264/SVC [1] codec. The video bit-stream is divided into Groups Of Pictures (GOPs) and quality layers, as shown in Figure 1. Moreover, the video file is divided into *chunks*, as for the standard BT.

This packet scheduling algorithm is similar to our solution presented in [8] and its objective is to maximise the received video bit-rate. A sliding window that consists of a fixed number of GOPs is defined. It contains the GOPs that follow the current playback position. The number of GOPs inside the window usually corresponds to 10-20 seconds of video. Inside the window chunks have a priority that depends on the quality layer they belong to. Layer q_0, which is also called the *base layer* needs to be completely received to decode a certain GOP. Therefore, it has the highest priority. All the other layers are *enhancement layers* and they can improve the received video quality. Inside each layer, rarest pieces are requested first. In contrast with our previous implementation [8], this also holds for the base layer. At regular intervals, the system performs a window shift and checks how many quality layers of the current GOP have been completely received. If at least the base layer has been received, it will be decoded, otherwise the GOP will be skipped.

The sliding window is also shown in Figure 1. In a) the peer has just joined the network. It calculates the current playback position from its neighbours and there is a short pre-buffering before the first window shift. Figure 1b) shows the final moments of the pre-buffering. The peer has already downloaded all the chunks it could afford to upload and it sends low-priority requests to its neighbours. After the window shifts in c), the completely received quality layers are decoded, while the partial ones are discarded.

3.2. Game Theoretic Framework

In our system, a credit line mechanism is used. However, as it was previously introduced, two different types of requests are defined: regular and low priority ones. This is necessary, as we prove that

peers should not *request data they cannot afford to upload*, but we would also like to exploit the spare resources in the system. We consider the case in which peers have different upload capacities and we use WSVC. Under these circumstances, *quality layers* are subsets of the original bit-stream associated to different SNR.

Theorem 1: Let us consider a peer P_i whose upload capacity is c_i and a layered scalable video whose bit-rates of layers $q_0, \ldots, q_{max}$ are $b_0, \ldots, b_{max}$. If this peer downloads data belonging to quality layers $q_j, \ldots, q_{max}$, such that $b_j > c_i$, it will be eventually marked as free-rider by its neighbours.

Proof: A peer P_i is marked as free-rider by its neighbours $P_k \in N_i$ if $\chi_k^s(i,t) - \chi_k^r(i,t) > \Delta_k^{\max}$ for any $k \mid P_k \in N_i$, where N_i is again the set of neighbours of P_i, $\chi_k^s(i,t)$ is the number of chunks P_k sent to P_i at time t, $\chi_k^r(i,t)$ is the number of chunks P_k received from P_i at time t and $\Delta_k^{\max}$ is the credit line set by peer P_k. Considering the most favourable case for peer P_i, which is when $\chi_k^r(i,t) = \chi_i^s(k,t)$ (e.g. there are no transmission errors), P_k will cooperate with P_i if $\lim_{t\to\infty}(\chi_i^s(k,t)/\chi_k^s(i,t)) = 1$. If this condition is not satisfied then, for an arbitrarily long time t', $\chi_k^s(i,t') - \chi_i^s(k,t') > \Delta_k^{\max}$ will be verified. As χ^s depends on the upload capacity of a certain peer, then a user should not try to download more data then it can afford to upload. Therefore, for the specific case of scalable video transmission, a peer should not aim at downloading the video with a bit-rate that is higher than its upload capacity. It also follows that free-riders and users with an upload capacity which is lower than bit-rate of layer q_0 will be cut off.

In some cases, however, there might be some spare resources in the system. Therefore, in order to fully exploit the available capacity, low priority requests are defined. These are requests that concern quality layers with a higher bit-rate than a peer can afford to upload. They will only be satisfied if the peer that receives one is not fully utilising its capacity and they will not be added to the current credit. Furthermore, is not convenient for a peer to ask all the chunks as low priority requests. For instance, no chunks belonging to the base layer can ever be requested under these circumstances, as it would imply that a peer does not meet the minimum requirements to be allowed inside the P2P network. Moreover, as far as enhancement layers are concerned, requests will only be satisfied if there are no high-priority ones in the buffer.

3.3. Delay Minimisation

The delay minimisation algorithm aims at achieving real-time video streaming through accurate resource allocation. In applications such as live streaming, it is fundamental to keep the upload channels of all the peers constantly busy. It has already been proven [9] that in order for the peers to be fully served, for a non-scalable video the following condition needs to be satisfied:

$$\sum_{i \mid P_i \in N} U_i \geq (|N| - 1) \times s \tag{3}$$

where N is the set of peers (including the source), U_i is the upload capacity of a peer P_i, $|N|$ is the total number of peers and s is the video bit-rate. It is possible to rewrite the condition stated in Equation (3) for scalable video sequences.

Lemma 1: Assuming that a scalable video with bit-rates $b_1, \ldots, b_j, \ldots, b_{max}$ is used, and each peer P_i has a target bit-rate b_j^i, in order to achieve these received bit-rates the following condition needs to hold:

$$\sum_{i \mid P_i \in N} U_i \geq \sum_{i \mid P_i \in N \setminus \{S\}} b_j^i \tag{4}$$

where S is the source of the video. Considering the total flow of data entering and leaving each peer, the Lemma follows. We now assume that the number of peers in the network is high and the contribution of the source is therefore negligible. Moreover, in order for a peer not to be marked as free-rider, following from the proof of Theorem 1 the outgoing flow of data on every single channel should roughly correspond to the incoming one. In this specific case, the former condition can be rewritten as $U_i \geq b_j^i,\ \forall i | P_i \in N \setminus \{S\}$ and $U_i = b_j^i$ only in case of optimal resource allocation. In some cases, however, this does not happen as some peers do not have any data their neighbours are interested in. This problem can be partially overcome uploading more data to the *most deprived* peers [4].

3.4. Optimal policy for delay minimisation and free-riding detection

As far as our resource allocation algorithm is concerned, a peer P_k adopts the following strategy when receiving a request r_i from P_i: every time a request is received, it is associated with a time-stamp and a time-to-live (TTL). Moreover, it is associated with a *score* $s(r_i)$, which is defined as follows:

- If $\chi_k^s(i,t) - \chi_k^r(i,t) > \Delta_k^{\max}$, $s(r_i) = -\infty$; this request will be immediately rejected, as P_i might be a free-rider.
- If $\chi_k^s(i,t) - \chi_k^r(i,t) \leq \Delta_k^{\max}$ and the request has been sent with high priority, the request will be inserted into a buffer and it will be associated with a score $s(r_i) = \left(\alpha \cdot \varphi_{dep}^i + \beta \cdot \varphi_{cr}^i\right)$, where φ_{dep}^i is the *deprivation factor* and φ_{cr}^i is the *credit factor* of P_i, and α and β are arbitrary constants which satisfy $\alpha + \beta = 1$. These quantities will be explained in detail in the following paragraphs.
- If $\chi_k^s(i,t) - \chi_k^r(i,t) \leq \Delta_k^{\max}$ and the request has been sent with low priority, the request will be inserted into the buffer with $s(r_i) = 0$.

Time is divided into rounds; at the end of each round, a peer accepts the request in its buffer which has the highest score, while the others remain until their TTL expires. The TTL depends on the arrival rate of the requests from its non free-riding neighbours. Its value is calculated to minimise the number of rounds with no requests in the buffer. If two or more requests have the highest score, the peer will upload the least forwarded chunk.

As far as the *deprivation factor* is concerned, it is calculated as follows. Considering all the non free-riding neighbours N_k^* of P_k, for every peer P_j in this set the number of chunks P_k could contribute to them are counted. This value is also called the *deprivation* of peer P_j, or $\delta(P_j)$. The deprivation factor φ_{dep}^i of a peer is therefore computed as:

$$\varphi_{dep}^i = \frac{\delta(P_i)}{\sum_{j \in N_k^*} \delta(P_j)} \tag{5}$$

In other words, φ_{dep}^i is the ratio between the number of chunks P_k has and P_i is missing and the total number of chunks the non-free-riding neighbours of P_k are missing. On the other hand, the *credit factor* φ_{cr}^i ranges in $(0, 1]$ and it indicates the contribution received by P_k from P_i with respect to the total contributions from the non free-riding neighbours of P_k. It is defined as:

$$\varphi_{cr}^i = \frac{\chi_k^r(i,t) - \chi_k^s(i,t) + \Delta_k^{\max} + 1}{\sum_{j \in N_k^*} \chi_k^r(j,t) - \chi_k^s(j,t) + \Delta_k^{\max} + 1} \tag{6}$$

The deprivation factor helps achieve nearly optimal performance, while the credit factor is used to encourage resource reciprocation.

3.5. Distribution of the First Copy

One of the critical aspects in P2P file sharing is the distribution of the first copy. In our approach, peers cannot request any data to the source, which only uploads data to selected users in the swarm. First of all, the source identifies the most deprived non-free-riding peer in the swarm. This can be achieved using a give-to-get (G2G) [10] like algorithm; depending on the information received from the peers in the swarm, the source will then classify users either as cooperative or deceptive. The second step consists of identifying the chunks this peer is missing. Among them, the source will then forward the one that has been forwarded the least amount of times.

4. RESULTS

We tested our algorithms using ns-2 simulator [11]. City sequence is repeated 15 times (for a total of 150 seconds) and encoded in WSVC format. The spatial resolution is 352×288 (CIF) and the frame rate is 30 fps. The video sequence is split into chunks of 4 KB. In addition, there are 7 quality layers, ranging from 256 kbps to 768 kbps. In our experiments, only one source has the sequence. Peers are divided into two categories: regular users and free-riders. Regular users follow the proposed algorithm, while free-riders reject any request they receive. The peers join the system within a one-second interval. After the pre-buffering, the playback starts. The size of the network is 31 or 51 peers. This includes a video source, which has the same bandwidth as the other users (75 KBytes/s). We consider scenarios with different fractions of free-riders (50% and 66%) and values of $\Delta^{\max}$. Moreover, α and β range between 0 and 1, with $\beta = 1 - \alpha$. A comparison with an existing technique [6] is also shown. In the next paragraphs, some selected results will be illustrated.

Figure 2 shows the impact of different $\Delta^{\max}$ on the behaviour of the system when there are 31 peers with 66% free-riders in the network and (α, β) is set to (0.5, 0.5). $\Delta^{\max}$ grows exponentially with base 2 and it ranges between 1 and 128 chunks. Tests are also performed with $\Delta^{\max} = \infty$. The average received video bit-rate grows as $\Delta^{\max}$ grows up to a certain threshold, which in this case is 4 chunks. For values below the threshold, cooperative peers are incorrectly detected as free-riders, which causes poorer performances of the system. On the other hand, using a $\Delta^{\max} > 4$ chunks causes a degradation in the received video bit-rate of cooperative users, while free-riders gain a benefit as the credit line grows. However, due to the G2G-like free-riding detection mechanism used by the source and the large amount of misbehaving peers in the network, their average received video bit-rate remains very low.

We are also considering the impact of α and β on the behaviour of the system. $\Delta^{\max}$ is set to 4 chunks and there are 51 peers in the network, 50% of which are free-riders. Figure 3 shows the results for cooperative peers when (α, β) varies between $(0, 1)$ and $(1, 0)$. An important remark is that for $(\alpha, \beta) = (0, 1)$ the system is very similar to the one presented in [6]. This graph indicates that when free-riders are in the swarm, considering both the credit and the deprivation factor gives better results than trying to solve the problem of resource reciprocation and delay minimisation alone.

5. CONCLUSION

We proposed a framework for P2P scalable video transmission that exploits elements from game theory and delay minimisation algorithms. Our simulation studies show that allocating resources considering the credit and the deprivation factors of a peer's neighbours

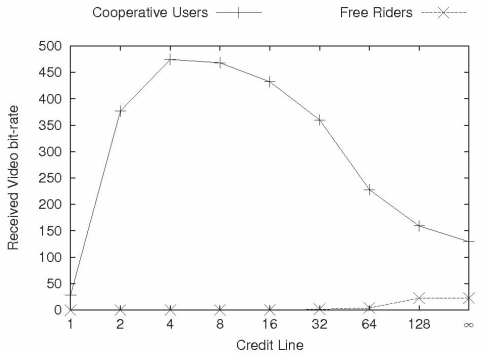

Fig. 2. Received video bit-rate with 31 peers and 66% free-riders.

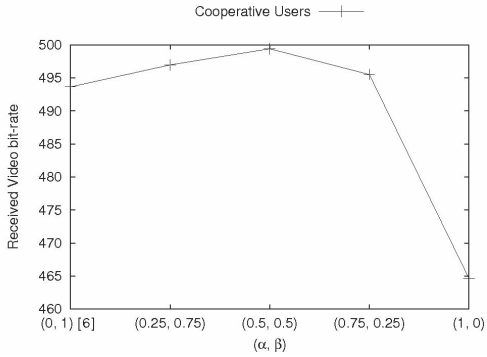

Fig. 3. Received video bit-rate for cooperative peers as a function of α and β with 51 peers and 50% free-riders.

gives better results than if only either of them is considered. Moreover, our experimental evaluation indicates that giving equal importance to both factors guarantees the best performance. Free-riders are cut out of the system, while cooperative users can achieve real time streaming.

As far as future work is concerned, we will consider different approaches, based on P2P network clustering. Furthermore, we will analyse the importance of social based features in such systems.

6. REFERENCES

[1] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the scalable video coding extension of the H.264/AVC standard," *IEEE Trans. Circ. and Sys.*, Sep. 2007.

[2] N. Ramzan, T. Zgaljic, and E. Izquierdo, "An efficient optimisation scheme for scalable surveillance centric video communications," *Signal Processing: Image Communication*, Jul. 2009.

[3] P. Baccichet, T. Schierl, T. Wiegand, and B. Girod, "Low-delay peer-to-peer streaming using scalable video coding," in *Proc. Packet Video 2007*, 2007.

[4] F. Picconi and L. Massoulie, "Is there a future for mesh-based live video streaming?" in *Proc. 8th Int. Conf. P2P Computing*, 2008.

[5] B. Cohen, "Incentives build robustness in BitTorrent," in *Proc. 3rd Int. Conf. P2P Computing*, Apr. 2003.

[6] W. S. Lin, H. V. Zhao, and K. J. R. Liu, "Incentive cooperation strategies for P2P live multimedia streaming social networks," *IEEE Trans. Mult.*, Apr. 2009.

[7] C. Buragohain, D. Agrawal, and S. Suri, "A game theoretic framework for incentives in P2P systems," in *Proc. 3rd Int. Conf. P2P Computing*, 2003.

[8] S. Asioli, N. Ramzan, and E. Izquierdo, "A novel technique for efficient P2P scalable video transmission," in *2010 European Sig. Proc. Conf.*, Aug. 2010.

[9] F. Huang, B. Ravindran, and A. Vullikanti, "An approximation algorithm for minimum-delay P2P streaming," in *Proc. Int. Conf. P2P Computing*, Sep. 2009.

[10] J. J. D. Mol, J. A. Pouwelse, M. Meulpolder, D. H. J. Epema, and H. J. Sips, "Give-to-get: free-riding resilient video-on-demand in P2P systems," in *Proc. SPIE, Multimedia Computing and Networking Conference (MMCN)*, 2008.

[11] "Network simulator 2," `http://nsnam.isi.edu/nsnam/index.php`.