On the Benefits of Synchronized Playout in Peer-to-Peer Streaming*

Constantinos Vassilakis
Dept of Informatics and Telecommunications
University of Athens, Greece
cvassilakis@noc.uoa.gr

Nikolaos Laoutaris
Dept of Informatics and Telecommunications
University of Athens, Greece
laoutaris@di.uoa.gr

Ioannis Stavrakakis
Dept of Informatics and Telecommunications
University of Athens, Greece
ioannis@di.uoa.gr

ABSTRACT

In this paper we examine the impact of the adopted playout policy on the overall performance of a P2P streaming system. It is argued and showed that adopting (popular) playout policies that result in a divergence of the playout points drastically deteriorates the performance of P2P streaming and that policies that keep these points "near-in-time" should be adopted.

Categories and Subject Descriptors

C.2.m [**Computer-Communication Networks**]: Miscellaneous

General Terms

Design, Performance

Keywords

Peer to Peer, Video Streaming, Playout Schemes

1. INTRODUCTION

We study *playout* and *peer selection* policies in peer-to-peer (P2P) systems for the dissemination of video streams [3]. Playout policies for video receivers have been studied extensively in the past for the client-server case, involving a single server and multiple, independently operating, receivers [2, 1]. P2P streaming systems, however, are fundamentally different. Besides rendering the received stream for the benefit of the local user, a receiver also acts as a sender and forwards it to other "downstream" receivers which, in turn, can forward it further down in a *hierarchy of peers*. In a client-server system each peer receives the stream directly from the sender and does not relay it any further. We argue that the new conditions have to be carefully factored-in when selecting playout policies for such systems, either when the same policy is adopted by each peer or when each peer autonomously selects its preferred policy. Overlooking them can easily lead to a totally unacceptable performance and the collapse of the system.

To exemplify the above point, we show that the *desynchronization of playout points* can have dire consequences on the probability of finding a better up-stream relay node when the current one is experiencing congestion or when it departs from the system without prior notice. We argue that keeping the playout points of different peers "near-in-time", is the right thing to do in a P2P setting. Keeping the playout points (nearly) synchronized creates "positive correlation" in terms of the contents of different playout buffers which, in turn, increases the availability of upstream relay peers to which a node can perform a fast, discontinuity free hand-off in times of poor reception from the current relay peer. A necessary prerequisite for achieving synchronization is to occasionally drop some "late" frames in order to catch-up with peers that have not experienced any lateness and, thus, are further ahead in time.

Assuming that nodes can tolerate some delay with respect to the video source (i.e., when there is no strict interactivity requirement), it is not at all obvious that dropping undisplayed frames makes sense. Indeed, late frames have already caused a "freeze" discontinuity due to buffer underflow. Discarding them only adds to the disruption by causing an additional "information loss" event (users would perceive that as a scene that initially freezes and then suddenly jumps ahead in time, skipping some of the ongoing activity). Displaying these frames avoids the information loss component of the overall disruption, and this is indeed the sensible thing to do in a client-server setting. In a P2P setting, however, such an approach backfires by causing desynchronization, as explained earlier. Our initial analytic and experimental results seem to indicate that such desynchronization is a much worse problem to handle than the occasional dropping of few late frames. Based on this realization we develop and evaluate synchronized playout schemes for use in P2P streaming applications and combine them with appropriate hand-off schemes of local, regional, or global information.

2. DEFINITION OF PLAYOUT SCHEMES

Let $e(n)$ denote the encoding time for the nth frame and $p_i(n)$ be its scheduled playout time at node v_i. We define the following playout schemes:

Sync(D_i): Frames that become available at peer v_i before their scheduled playout time are displayed at their exact playout time p_i. Frames that miss their playout time are skipped. This amounts to synchronous playout between the source and node v_i where by *synchronous* we indicate a *fixed*

*Work supported in part by the IST-FET project CASCADAS (027807) and the IST NoE CONTENT (0384239) funded by the European Commission.

CONEXT'06 December 4-7,2006, Lisboa, Portugal
Copyright 2006 ACM 1-59593-456-1/ 06/ 0012 ...$5.00.

offset between encoding and playout times. That is: $p_i(n) = e(n) + D_i$. When $D_i = D$, $\forall v_i \in V$ all nodes see a frame at the exact same time and a global sychronization is achieved.

Async(D_i): A frame gets displayed at the earliest possible time following the previously displayed frame. Assuming that all frames are delivered, we can define Async recursively as follows: $p_i(n) = p_i(n-1) + T + I_{\{b_i(n-1)<1\}} \cdot U(n-1), p_i(1) = e(1) + D_i$, where T is the duration of a frame, $b_i(n-1)$ is the number of frames in the buffer of v_i after the presentation of frame $n-1$, $U(n-1)$ is the duration of a possible underflow that follows the presentation of frame $n-1$, and $I_{\{\}}$ is the indicator function.

3. BASIC SYSTEM OPERATION

We assume a typical P2P streaming scheme as in [3] in which peers form a hierarchy rooted at the single video source. Our examination of playout schemes is orthogonal to the employed video encoding, therefore we assume single layer encoding for the sake of simplicity. Based on such a setting, we prescribe how peers join the hierarchy, select parents, and perform hand-offs.

New Node Join: Let $C_i(t)$ denote the *credit* of peer u_i at time t, i.e., its remaining discontinuity-free playout time if its input rate fall to zero at time t. The playout buffer can be thought to be draining from the bottom (position 1 holding the currently displayed frame) and filling from the top (the most recently received frame being at position b_i, which is also the buffer occupancy at time t). Let also $id\{x\}$ denote the id of the frame at position x of the buffer. Then we can define the credit as follows: $C_i(t) = p_j(id\{b_j\}) + T - t$.

If at time t the new node is v_i and its parent is v_j[1] then v_i starts receiving from v_j the frame at buffer position $x, 1 \leq x \leq b_j$ and all subsequent ones and starts displaying them $p_j(id\{x\}) - t + D_i - D_j(t)$ time units after the connection time t[2 3]. Thus $p_i(id\{1\}) = p_j(id\{x\}) + D_i - D_j(t)$. We set $x = b_j$ which amounts to retrieving from u_j its newest frame and all subsequent ones [4].

Parent Selection Strategy: Our initial results presented later assume a *random parent selection strategy*. Most implemented systems use this strategy, mainly due to its simplicity (we have also considered selection strategies which make use of partial or global information about the group, but do not report such results here). We allow a node v_i to be in either of the following two modes:

Stable mode: Node v_i is stably connected to its parent $f(v_i)$ as long as its buffer occupancy b_i is above a threshold B_h.

Handoff mode: Node v_i enters a handoff mode as soon as its buffer occupancy falls beneath B_h. The handoff mode includes the following steps:

1. Selection of a new parent by picking a node uniformly at random from the set $(V - \{v_i, f(v_i)\})$.

2. Connection to the new father for a "grace period" T_g and then return to the stable mode.

[1] v_i and v_j are considered to have synchronized clocks.

[2] $D_j(t)$ denotes the offset between encoding and playout times at time t to node j. For the sync case $D_j(t)$ is constant $\forall$ t.

[3] For $D_i < D_j(t)$ i.e. u_i is less interactive that its parent the received frame will be presented at u_i, $D_j(t) - D_i$ earlier than at u_j while for $D_i > D_j(t)$ the received frame will be presented at u_i, $D_i - D_j(t)$ later than at u_j. For $D_i = D_j(t)$ it will be presented to both the same time.

[4] This way the period for buffer buildup subject to the targeted D_i is maximized, giving the chance to prefetch the largest number of frames into the buffer while achieving D_i.

Performing the Handoff: Node v_i needs to get from its new parent $v_j = f(v_i)$ all frames with ids greater than $id\{b_i\} + 1$ at link speed. If not all of these frames are available at v_j then v_i starts receiving the ones that exist and skips the ones that are missing.

4. EVALUATION

We compare Sync(D) and Async(D) based on the following performance metrics:

Discontinuity: Under Sync, the discontinuity increases by T with each frame that misses its scheduled playout time. Under Async, the discontinuity increases with each underflow, by an amount that equals the duration of the underflow.

Loss: Under both Sync and Async, each frame that is not displayed increases the loss by T.

Next we briefly state a simulation experiment from which we draw some indicative results. We assume 10 peers which enter the streaming hierarchy closely in time one after the other. We assume that these peers participate in the system for the duration of our observation and that they do not suffer from buffer overflows. The frame period is $T = 1/30$ seconds, the offset is $D = 150 \cdot T$, the buffer threshold is $B_h = 30$ frames and the grace period is $T_g = 90 \cdot T$. The video source at the root of the delivery tree makes available a new frame every T seconds. Frame sizes are retrieved from a trace file of an educational video encoded at constant bit rate of 256Kbps. We consider that at each time slot of T seconds an overlay link L_i is "down" with a probability P_i. P_i is defined at start $\forall$ i after a random permutation of overlay links and distribution of the probability according to a generalized power law with parameter α. At each time slot that an overlay link is "up" the exact value of the transmission rate is drawn uniformly from the range [10,1500] Kbps. By using different a we model different levels of heterogeneity in terms of expected overlay link rates. We also employ a weight W to capture the overlay network's congestion level.

TABLE 1

	Sync(D)				Async(D)			
α	W=50		W=70		W=50		W=70	
	d(%)	l(%)	d(%)	l(%)	d(%)	l(%)	d(%)	l(%)
0.01	9.6	9.6	32.7	32.7	28.4	34.6	77.7	79.2
0.3	0	0	26.1	26.1	0	0	72.1	73.9
0.7	0	0	15.9	15.9	0	0	54.6	55.6
1	0	0	2.3	2.3	0	0	13.4	13.6

Let d denote the average discontinuity ratio expressed as the average among all peers of the total time that a peer spent viewing some frozen frame to the total playback time. Let l denote the average loss ratio expressed as the average among all peers of the total lost playback time experienced by a peer to the total playback time of all frames that should be presented to the user. First results (Table 1) indicate that d and l in Async(D) case are at least 2.4 times greater than the ones observed in Sync(D) case under the same conditions, in several conducted experiments with various congestion levels and heterogeneity values. In both cases it is observed that discontinuity and loss decrease as heterogeneity increases.

5. REFERENCES

[1] N. Laoutaris, B. V. Houdt, and I. Stavrakakis. Optimization of a packet video receiver under different levels of delay jitter: An analytical approach. *Performance Evaluation*, 55(3-4):251–275, 2004.

[2] N. Laoutaris and I. Stavrakakis. Intrastream synchronization for continuous media streams: A survey of playout schedulers. *IEEE Network Magazine*, 16(3), 2002.

[3] S. Rao. Establishing the viability of end system multicast using a systems approach to protocol design. *Phd Thesis, Technical Report CMU-CS-04-168, Carnegie Mellon University*, Oct. 2004.