Priority-Progress Streaming for Quality-Adaptive Multimedia*

Charles Krasic†
Oregon Graduate Institute
20000 NW Walker Rd.
Beaverton, Oregon 97206
krasic@cse.ogi.edu

Jonathan Walpole‡
Oregon Graduate Institute
20000 NW Walker Rd.
Beaverton, Oregon 97206
walpole@cse.ogi.edu

ABSTRACT

The Internet's ubiquity amply motivates us to harness it for video distribution, however, its best-effort service model is in direct conflict with video's inherent timeliness requirements. Today, the Internet is unrivaled in its rich composition, consisting of an unparalleled assortment of networks and hosts. This richness is the result of an architecture that emphasizes interoperability over predictable performance. From the lowest levels, the Internet architecture prefers the best effort service model. We feel current solutions for media-streaming have yet to adequately address this conflict between timeliness and best-effort service.

We propose that streaming-media solutions targetted at the Internet must fully embrace the notion of graceful degradation, they must be architected with the expectation that they operate within a continuum of service levels, adjusting quality-resource trade-offs as necessary to achieve timeliness requirements. In the context of the Internet, the continuum of service levels spans across a number of time scales, ranging from sub-second timescales to timescales as long as months and years. We say sub-second timescales in relation to short-term dynamics such as network traffic and host workloads, while timescales of months and years relate to the continuous deployment of improving network, compute and storage infrastructure.

We support our thesis with a proposal for a streaming model which we claim is simple enough to use end-to-end, yet expressive enough to tame the conflict between real-time and best-effort personalities of Internet streaming. The model is called Priority-Progress streaming. In this proposal, we will describe the main features of Priority-Progress streaming, which we have been implemented in a software-based streaming video system, called the Quasar pipeline.

Our work is primarily concerned with the class of streaming applications. To prevent confusion, we now clarify the important distinction between streaming and other forms of distribution, namely download. For a video, we assume download is defined so that the transfer of the video must complete before the video is viewed. Transfer and viewing are temporally sequential. With this definition, it is a simple matter to employ Quality-adaptive video. One algorithm would be to deliver the entire video in the order from low to high quality components. The user may terminate the download early, and the incomplete video will automatically have as high quality as was possible. Thus, Quality-adaptive download can be implemented in an entirely best-effort, time-insensitive, fashion. On the other hand, we assume streaming means that the user views the video at the same time that the transfer occurs. Transfer and viewing are concurrent. There are timeliness requirements inherent in this definition, which can only be reconciled with best-effort delivery through a time-sensitive adaptive approach.

1. PRIORITY-PROGRESS STREAMING

The central notion of Priority-Progress streaming is to decompose application data into units of work, application data units (ADUs), each labeled with timestamp and priority. The timestamp is meant to capture the timeliness requirements of each ADU, and is expressed in units of the *normal play time* of the media stream. As ADUs are processed end-to-end, the timestamps and priorities provide vital information necessary to regulate work so as to ensure proper real-time progress. The priority exposes the layered nature of the media, where quality can be progressively improved given more of the limiting resource: network, processing, or storage.

We use the priorities to achieve graceful degradation. Although a threshold priority-drop approach could be applied rather directly to match quality to available resources, there remains an issue that the resource-quality relationship may vary rapidly, and a user will likely be annoyed by the unstable quality. Our own experience implementing a Quality-Adaptive video system has shown us that streams can have non-smooth and highly dynamic quality-rate relationships, which are inconsistent across resource types[3]. Furthermore, the experience of others indicates to us that predict-

*This work was partially supported by DARPA/ITO under the Information Technology Expeditions, Ubiquitous Computing, Quorum, and PCES programs and by Intel.

†Phd student

‡Phd advisor

ing available network bandwidth is equally problematic[4]. We are thus motivated to reformulate the problem to avoid explicitly predicting either priority-rate relationship or resource availability. The unique aspect of Priority-Progress streaming presented here is that it uses ADU-reordering in its buffers to do just that.

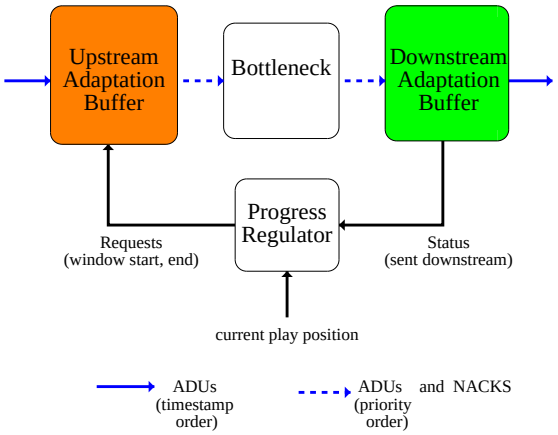

Figure 1: Priority-Progress Control

Figure 1 depicts the structure of Priority-Progress control within a media pipeline. A pair of re-ordering buffers is employed around each *bottleneck* pipeline component. For example, in the Quasar pipeline we have one such element responsible for streaming across a network transport. Similarly, the software-decompression element is considered a bottleneck, as it has unpredictable progress rates due both to data dependencies in MPEG and to external influences from competing tasks in a multi-tasking environment. The capacities of the re-ordering buffers are managed in terms of time, making use of the timestamp labels on ADUs[1]. The algorithm for Priority-Progress Streaming contains three subcomponents, for the upstream buffer, downstream buffer, and progress regulator respectively.

The upstream and downstream buffer algorithms operate as follows. The upstream buffer admits all ADUs within the time boundaries provided by the progress regulator, these boundaries delimit the *adaptation window*. Each time the regulator advances the window forward, the unsent ADUs from the old window position are expired and the window is populated with ADUs of the new position. ADUs flow from the buffer in priority-order through the bottleneck to the downstream adaptation buffer, as fast as the bottleneck will allow. The downstream adaptation buffer collects ADUs and re-orders them to timestamp order. ADUs are allowed to flow out from the downstream buffer when it is known that no more ADUs for a timestamp are coming.

To explain how the progress regulator works, it is important to understand how the flow of ADUS relates to the presentation timeline, as shown in Figure 2. The timeline is based on the usual notion of *normal play time*, where a presentation is thought to start at time zero (epoch a) and run to its duration (epoch e). Once started, the presentation

[1]For simplicity, we consider the buffers unbounded in terms of space, although space constraints can be enforced without difficulty.

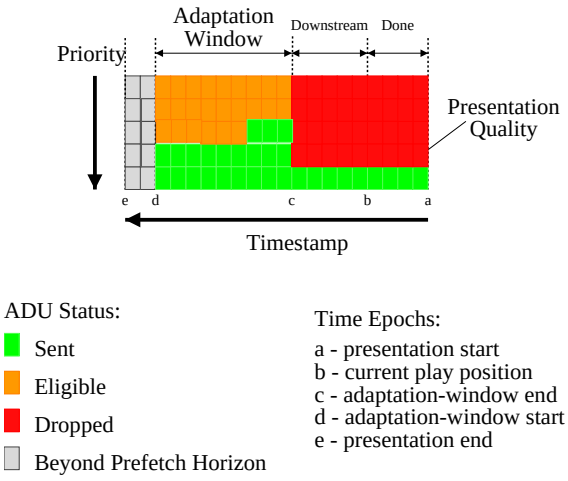

Figure 2: Priority-Progress Timeline

time (epoch b) advances at some rate synchronous with real-time. The ADUs within the adaptation window in the timeline correspond to the contents of the upstream and downstream re-order buffers; ADUs within the adaptation window that are *sent* are either in the bottleneck or the downstream buffer, while ADUs still *eligible* are in the upstream buffer. The interval spanned by the adaptation window is the key to our ability to control the responsiveness-stability trade-off of quality adaptation. The larger the interval, the less responsive and the more stable quality adaptation will be. A highly responsive system is generally required at times of interactive events (start, fast-forward, etc.), while stable quality is generally preferable. We transition from responsiveness to stability by progressively expanding the adaptation window. The regulator can manipulate the size of the window through actuation of the ratio between the rate at which the adaptation window is advanced and the rate at which the presentation clock advances. By advancing the timeline faster than the presentation clock (ratio > 1), the regulator can expand the window with each advancement, skimming some current quailty in exchange for more stable quality later.

2. REFERENCES

[1] W. chi Feng, M. Liu, B. Krishnaswami, and A. Prabhudev. A priority-based technique for the best-effort delivery of stored video. In *SPIE/IS&T Multimedia Computing and Networking 1999*, San Jose, California, January 1999.

[2] N. Feamster, D. Bansal, and H. Balakrishnan. On the interactions between layered quality adaptation and congestion control for streaming video. In *11th International Packet Video Workshop (PV2001)*, Kyongiu, Korea, April 2001.

[3] C. Krasic and J. Walpole. QoS scalability for streamed media delivery. CSE Technical Report CSE-99-011, Oregon Graduate Institute, September 1999.

[4] R. Rejaie, M. Handley, and D. Estrin. Quality adaptation for congestion controlled video playback over the internet. In *Proceedings of ACM SIGCOMM '99 Conference*, Cambridge, MA, October 1999.