

Control Challenges in Multi-level Adaptive Video Streaming

Dylan McNamee, Charles Krasic, Kang Li, Ashvin Goel,
 Erik Walthinsen, David Steere and Jonathan Walpole
 Oregon Graduate Institute of Science and Technology
 Department of Computer Science and Engineering
 Portland, Oregon

Abstract

Streaming video is one of the fastest-growing applications of the Internet. The Internet’s diversity and dynamism demands that video streams adapt to ensure maximum quality at all times. This paper describes the control challenges we have encountered in the Quasar project’s “multi-level” adaptive streaming video player. We first describe the framework and environment of the player. This framework uses software feedback to control resource allocation as well as the quality of media delivery. We present the control challenges raised by our framework, which include horizontal and vertical feedback composition, difficult to model systems, and unpredictable, non-linear actuators. We describe some of the approaches we are taking to address these challenges, related work, and future application areas and the challenges they will raise.

1 Introduction

Streaming video is one of the fastest growing applications of the Internet. In spite of this, current video streaming systems do not gracefully accommodate the Internet’s primary attribute—heterogeneity. The heterogeneity of the Internet includes clients and servers of widely differing capacities as well as diverse and dynamic network connections between them. This diversity causes the amount of resources available between video servers and clients to vary, both from installation to installation, and dynamically at a single installation. If insufficient resources are available anywhere along the video pipeline, quality rapidly degrades to unacceptable levels. One approach to avoid this outcome is to have the applications use reservations to ensure enough resources are available (e.g., RSVP [14] and RT-Mach [7]). When reserving all of the resources from a server through the network to a client is possible, this approach can work. However, if *any* link in a reservation-based system is shared with another application or does not support reservations, unacceptable degradation is inevitable. To address this common case, the

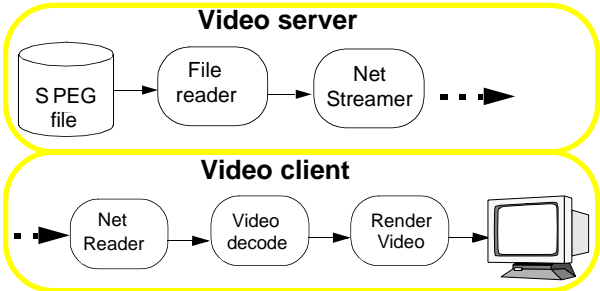


Figure 1: A distributed video pipeline.

Quasar project is exploring an approach in which applications use feedback control to dynamically adapt to the amount of resources available.

The Quasar adaptation framework is based on pipelines of processes. For a distributed video player, the processes correspond to the stages of reading the video from disk, sending it over the network, receiving it from the network, decoding, and displaying the decoded video. Figure 1 depicts a simple distributed video pipeline.

Our system uses feedback controllers to actuate a wide range of system interfaces, from resource allocation to application adaptation. The control challenges include monitoring and actuation interfaces, the design and behavior of the

This project was supported in part by DARPA contracts/grants N66001-97-C-8522, N66001-97-C-8523, and F19628-95-C-0193, and by Tektronix, Inc. and Intel Corporation.

individual feedback controllers as well as the interaction between them.

The rest of this paper is structured as follows: Section 2 presents the various levels of feedback control in our multi-level adaptive video player. For each level, we describe the mechanism used by the actuators, the goal and structure of the feedback controller, and the challenges we encountered in design and implementation. Section 3 relates various aspects of our work to other research in the area. Finally, Section 4 summarizes and concludes.

2 Feedback control of an adaptive media pipeline

The Quasar pipeline uses feedback to control adaptation at three levels. The first level adjusts the resources allocated to each pipeline element (e.g., CPU and network bandwidth). The second level adjusts the resources required by the application by adapting the presentation quality. The third level adjusts the application adaptation policy to adapt to new environments or user requirements.

This section describes each level of adaptation in turn, its actuation and monitoring interfaces, its controllers, and the control challenges it poses.

2.1 Feedback control of resource allocation

Each element of the pipeline is a consumer of one or more system resources. For example, the *Video decoder* consumes client CPU, and the *Net streamer* consumes both CPU and network bandwidth on the server. One possible bottleneck in the system is insufficient allocation of one of these resources to any element of the pipeline.

In contrast to approaches that use reservations to solve the resource allocation problem statically, our approach dynamically adjusts allocation according to dynamically observed requirements. This is accomplished by the first level of control, which allocates resources to each pipeline element in order to meet the rate requirements of the pipeline. The CPU and network bandwidth controllers have as their goal to match the bandwidth required by the video

stream. The overall rate of the pipeline is determined by a pipeline element with external timing requirements. For example, the pipeline's rate could be driven by the client's renderer being clocked at 30 fps, or by the server side reading and streaming data at 30 fps. Without such a *real-rate driver*, the pipeline could acceptably run at any rate, including stopped.

Each resource controller determines the rate requirements of each element it schedules by monitoring buffer fill-levels on either side of the element. The controller actuates a resource manager, which exports a proportion/period interface for the resource. For example, if the buffer fill level going into the *Network streamer* element is falling, then the controller will reduce the CPU proportion allocated to it. Alternatively, if the buffer fill level between the *Network streamer* and the network device is rising, then that controller will allocate more network bandwidth to the stream.

A significant challenge in the design of the monitoring aspect of the pipeline is the choice of units of monitoring. Two useful alternatives are to monitor progress in terms of data size (e.g., bytes or packets), or to monitor progress in terms of application-level time. Monitoring sizes is a simple, application-independent mechanism, but it may not be an accurate representation of the actual rate of progress, especially considering the impact of variable bit-rate compression. In addition, size-based metrics complicate reasoning about a pipeline's latency. To address these problems, we have developed an alternative approach based on monitoring progress in terms of application-level time. For example, a buffer could hold 330ms of video frames, which would be ten frames at 30 frames per second, or five frames at 15 frames per second. This approach solves the problems with variable bit-rate streams since the amount of time represented by the buffers is independent of the bit rate. A challenge posed by this approach is to develop clean interfaces that preserve the semantic separation of the application from the underlying system.

The individual controllers at this level were constructed using the Software Feedback Toolkit toolkit (SWiFT) [1], and are summarized by another paper submitted to this session [10]. The

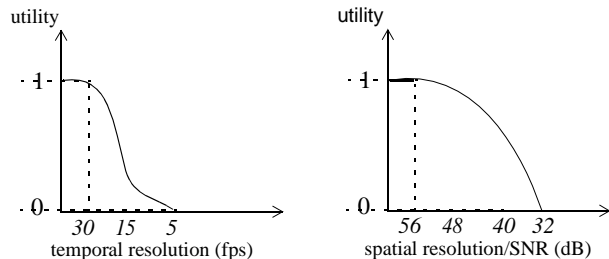


Figure 2: Utility functions for spatial and temporal resolution.

remainder of this paper draws on their presentation as needed.

2.2 Feedback control of presentation quality

If any resource manager in the pipeline is unable to meet the pipeline’s real-rate requirements, the quality of the presentation can be reduced in order to fit in the available amount of resources. We have developed a model for describing and expressing this kind of adaptation, based on quality axes, utility functions, and an adaptation model based on data-dropping.

2.2.1 Quality adaptation mechanism: SPEG

A *quality axis* is a media attribute that can be independently adapted. Users express their adaptation policy preferences by assigning utility values to the possible values for each quality axis. The utility values express the utility of minimum and maximum quality thresholds, as well as the relative importance of specific adaptation values for each axis.

We have implemented a quality-scalable version of the MPEG-1 format, called SPEG [3]. This format separates the adaptation axes of temporal resolution (frame-rate) from spatial resolution (signal-to-noise). Figure 2 shows an example set of utility functions for these axes. The x-origin of each function represents perfect quality, with increasing values corresponding to increased presentation error. The y-axis normalizes user utility between 0 (useless) and 1 (indistinguishable from perfect). The useful range of adaptation in the system for each axis lies between the projections of the utility values of one and zero. Reducing quality beyond the level deemed to be useless is not considered, nor is

improving quality above the level deemed indistinguishable from perfect.

The data in an SPEG stream is framed so that the data for any one quality dimension can be dropped without affecting other quality axes. A *quality mapper* translates the utility functions into priority labellings of the SPEG packets. Media quality is adapted by dropping data whose priority labelling is below a set threshold. The media quality actuator consists of increasing or decreasing the dropping threshold.

2.2.2 Automatic quality adaptation control

An outstanding challenge in our system is to integrate the resource-level feedback controllers with the quality adaptation mechanism. The goal of this integration is for the resource-level controllers to recognize bottlenecks in the system and respond by actuating presentation-quality adaptation. We have considered a number of options, including fill-level thresholds, resource manager exceptions, transitive communication, and global communication.

One way to detect a bottleneck that will ultimately affect presentation quality is that the buffer fill level between any two stages becomes totally full or totally empty. These cases are not symmetric, and whether they require quality adaptation depends on whether the real-rate driver of the pipeline is a data-source or a data-sink. If the driver is a data-sink (e.g., a video display), empty buffers indicate impending presentation degradation. Alternatively, if the driver is a data-source (e.g., a video capture device), full buffers indicate impending data loss due to buffer overflows. A buffer-based approach to triggering quality adaptation is to set low- or high- “water marks” on fill levels, which when reached would automatically adapt presentation quality. One drawback of this approach is that buffers need to be appropriately sized, and the water mark-thresholds set so that these events happen early enough to adapt quality before data is missed, yet not so early as to adapt too often.

An alternative adaptation is based on monitoring utilization of the resource managers. If any resource manager in a pipeline detects overload, then eventually a pipeline stage controlled by that manager will fall behind its real-rate require-

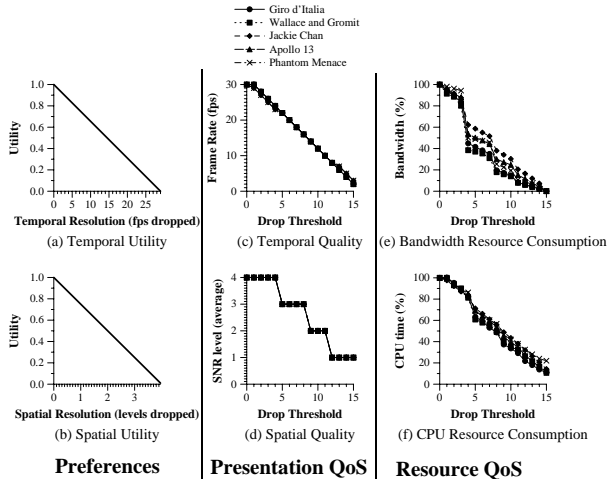


Figure 3: Quality-level vs. resource-level impact of actuating quality adaptation for linear utility functions.

ments. When this occurs, the resource manager can signal an application under its control to reduce its presentation quality.

Presentation quality can be adapted by any stage in the pipeline by discarding data in priority order. When quality has been adapted at one pipeline stage, the higher quality data processed “upstream” of that stage is, in effect, wasted. Thus such quality adaptations are transitively propagated toward the data source. An alternative, which results in more responsive adaptation, is to transmit this information out-of-band in a global communication.

2.2.3 Media adaptation control challenges

The first control challenge we have encountered at the media adaptation layer is the effectiveness of the quality adaptation actuators, as expressed in SPEG packet priority levels. Figure 3 shows the utility functions, quality levels, and resulting resource consumption levels produced by our current quality mapper for simple, linear utility functions. The quality mapper assigns priority to packets in order to affect a linear degradation of presentation quality along each axis. In this example, preference was given to neither quality axis, so adaptation proceeded uniformly among them. The step-function of the spatial quality is a result of our SPEG encoding, which provides four levels of SNR quality. The challenge in the design of this actuator is that uniform adjustment of dropping threshold should result in linear

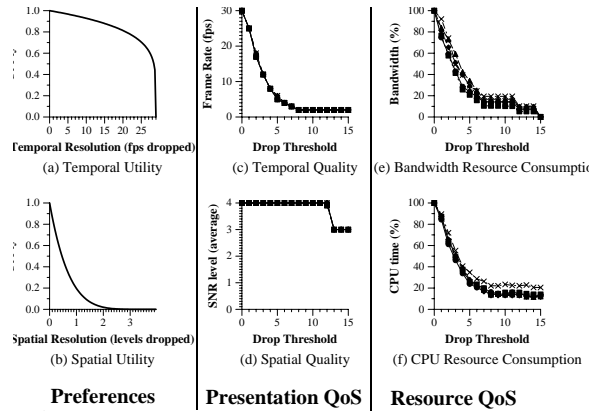


Figure 5: Quality-mapping impact of nonlinear utility functions.

changes in resource consumption. Figure 4 shows the presentation and resource levels that result from more interesting utility functions. We found that the nonlinearity of the resource consumption curves pose problems for feedback circuits that automatically actuate this mechanism. We are exploring quality mapping algorithms that make the resource consumption profiles linear, independent of the utility functions. This task is non-trivial, because it depends greatly on the content of the stream. For a stored movie, off-line computation can achieve near linearity of actuation. For interactive video, achieving linearity will be more challenging, and may require another feedback controller that monitors the profile of the labelled stream, and adjusts the labelling policy when the monitored profile deviates from linear.

The other challenge for presentation-quality adaptation is its interaction with the resource-level feedback controllers. This interaction raises a number of issues. First, the frequency of monitoring and actuation of the various controllers need to provide stable behavior. Second, quality-level adaptations can produce significant disturbances in the resource-level feedback systems. For the first challenge, we are investigating modeling and verification analysis of the feedback controllers [10]. For the second challenge, we have explored using explicit communication between the resource-level and quality-level controllers to “jump-start” them into configurations previously known to work for the new configuration. For example, we implemented this technique on a laptop system to quickly adapt between high- and low- quality when switching

between ethernet and modem connections [2]. This system quickly adapts quality up as well as down. Without such explicit resource-level notifications, we have found adapting quality upward to be a significant challenge, since it is difficult to detect when adapting upward will not immediately result in a downward adaptation. In particular, such quality oscillations are undesirable to users.

We plan to explore other adaptation axes in the future. Some of these axes are media-specific, such as adjusting audio quality in a multimedia stream. Other axes are independent of the stream's content, such as allowing tradeoffs between a stream's jitter and its latency. This level of trade-off is enabled because progress is measured in terms of time instead of bytes. A pipeline's latency can be reduced by shrinking the time contained by the buffers, at the expense of higher jitter. This kind of trade-off would be useful for a video conferencing system, and could be made dynamically.

Another important aspect of the interaction between resource-level and presentation quality adaptation is the impact of choice of buffer size on responsiveness and stability, in addition to its impact on latency and jitter. Larger buffers reduce the responsiveness required of resource-level adaptation in order to avoid buffer under- or over-flow, as well as the reducing the responsiveness required of presentation-quality adaptation.

2.3 Feedback control of adaptation policy

The third level of control is adjusting the adaptation policy. Examples of this kind of adjustment is to change the labelling policy performed by the quality mapper (e.g., using different utility functions for the quality axes). This level of control is driven by changes in the system environment. For example, if the end-to-end system bottleneck was originally network bandwidth, but subsequently changed to client CPU, a useful reaction would be to relabel the stream in order to provide linear actuation of the CPU resource. The main challenge in this scenario is to accurately identify the overall system bottleneck (client, server, network), and the resource shortage at that location which is causing the bottleneck (e.g., CPU or network bandwidth). A mistake in

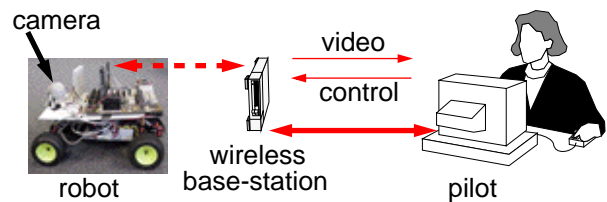


Figure 6: Tele-operated vehicle platform for exploring dynamic environments

this identification will result in worse presentation quality than not adapting at all.

We have built an experimental platform to explore the second example – automatic adaptation to dynamic environments. The platform consists of a mobile robot that incorporates a computer running a modified version of the Linux operating system. The computer is intended to represent the kinds of embedded systems that will be available in four to eight years. It has no disk, but does have a wireless Internet connection and a video camera. Figure 6 shows the configuration of the vehicle, camera, network and remote pilot. The remote user pilots the robot by viewing the video stream captured by the robot's camera.

The motivation for adjusting the adaptation policy in this platform is that as the vehicle's speed increases, higher frame-rate (temporal resolution) enables more accurate control for navigation. Third-level adaptation automatically controls the adaptation policy to adjust the relative utilities of the quality dimensions based on the vehicle's speed. When the vehicle is at rest, as the available resources vary, the video pipeline obeys the default quality adaptation policy, which is defined by the set of utility functions. A controller on the vehicle monitors its speed and assigns increasing weight to the temporal resolution utility function proportional to the vehicle's speed. This new set of utility functions is used to adapt the video to the current level of available resources, but with a policy that favors frame rate over resolution as speed increases.

3 Related work

Adaptive multimedia streaming is an active area of research. Our approach is novel in its architecture based on multiple levels of control to adapt to different aspects of dynamic environments. Our resource-level adaptation is an extension of

the use of feedback in the Synthesis system [5], and is related to other feedback-based resource allocation systems [4]. Our proportion/period based actuation interface is related to a number of efforts to implement proportional allocation [11, 12].

Our second- and third-levels of adaptation are related to systems that involve the application in adaptation, for example in mobile systems [8]. Our video-specific adaptation is related to other methods of video adaptation, including layered multicast [6], QoS filters [13], and the QoS Resource Allocation Model [9].

4 Summary and conclusions

Streaming dynamic media over the Internet requires a broad range of adaptations. We have presented the control challenges posed by the Quasar adaptive streaming multimedia system. The decentralized approach we use distributes control over the system's adaptation. Feedback controllers monitor and actuate three levels of the system: resource allocation, application quality adaptation policy, and dynamic control of the adaptation policy. Each level monitors aspects of the system that impact that level, and locally actuates the system in response. We described the overall system and discussed the issues raised by each level, including the challenges related to monitoring, actuation and control, as well as cross-level interactions raised by our system's architecture.

5 Acknowledgements

The SPEG converter was co-designed and implemented by Mark Jefferys. Discussions with Anne-Françoise Lemeur and Dave Maier contributed to the ideas in this paper.

6 References

[1] Goel, A., D. Steere, C. Pu, and J. Walpole. *SWiFT: A Feedback Control and Dynamic Reconfiguration Toolkit*. CSE-98-009, Oregon Graduate Institute. Portland, OR. 1998.

[2] Inouye, J., J. Binkley, and J. Walpole. *Dynamic Network Reconfiguration Support for Mobile Computers*. in *Third ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom '97)*. 1997. Budapest, Hungary.

[3] Krasic, C. and J. Walpole. *QoS Scalability for Streamed Media Delivery*. CSE-99-011, Oregon Graduate Institute of Science and Technology.

Beaverton, Oregon. 1999.

[4] Lu, C., J. Stankovic, G. Tao, and S. Son. *Design and Evaluation of a Feedback Control EDF Scheduling Algorithm*. in *Real-Time Systems Symposium*. 1999.

[5] Massalin, H. and C. Pu, *Fine-Grain Adaptive Scheduling Using Feedback*. *Computing Systems*, 1990. **3**(1): p. 139-173.

[6] McCanne, S., M. Vetterli, and V. Jacobson, *Low-complexity Video Coding for Receiver-driven Layered Multicast*. *IEEE Journal on Selected Areas in Communications*, 1997. **16**(6): p. 983-1001.

[7] Mercer, C.W., S. Savage, and H. Tokuda. *Processor Capacity Reserves: Operating System Support for Multimedia Applications*. in *Proc. of the International Conference on Multimedia Computing and Systems*. 1994.

[8] Noble, B., M. Satyanarayanan, D. Narayanan, J.E. Tilton, J. Flinn, and K. Walker. *Agile Application-Aware Adaptation for Mobility*. in *Symposium on Operating System Principles*. 1997. Saint-Malo, France.

[9] Rajkumar, R., C. Lee, J. Lehoczky, and D. Siewiorek. *A Resource Allocation Model for QoS Management*. in *The IEEE Real-Time Systems Symposium*. 1997.

[10] Steere, D.C., M.H. Shor, and J. Walpole. *Control and Modeling Issues in Computer Operating Systems: Resource Management for Real-Rate Computer Applications*. submitted for publication as part of this invited session. 2000. CDC00-INV5704.

[11] Stoica, I., H. Abdel-Wahab, and K. Jeffay. *On the Duality Between Resource Reservation and Proportional Share Resource Allocation*. in *Multimedia Computing and Networking*. 1997. San Jose, CA.

[12] Waldspurger, C.A. and W.E. Weihl. *Lottery Scheduling: Flexible Proportional-Share Resource Management*. in *Symposium on Operating Systems Design and Implementation (OSDI)*. 1994.

[13] Yeadon, N., *Quality of Service Filters for Multimedia Communications*, . 1996, Lancaster University. Lancaster, UK.

[14] Zhang, L., S. Deering, D. Estrin, S. Shenker, and D. Zappala, *RSVP: A New Resource Reservation Protocol*. *IEEE Network*, 1993. **7**: p. 8-18.