Quality of Service Specification for Multimedia Presentations*

Richard Staehli, Jonathan Walpole and David Maier
{*staehli, walpole, maier*}*@cse.ogi.edu*

Department of Computer Science & Engineering
Oregon Graduate Institute of Science & Technology
20000 N.W. Walker Rd., PO Box 91000
Portland, OR 97291-1000

ABSTRACT

The usefulness of a multimedia presentation depends on the accuracy of its output values and the timing of those outputs. A Quality of Service (QOS) specification is a vehicle for requesting accuracy guarantees from a multimedia system. This paper gives a formal model for presentation-level QOS specification that constrains presentation outputs only. Such a QOS specification leaves a multimedia system free to optimize resource management while providing end-to-end guarantees for multimedia services. An error model is proposed with a complete set of QOS parameters for specifying presentation quality. We show how this error model extends the opportunities for optimizing resources within a multimedia system.
Keywords: Quality of Service, Multimedia, Synchronization, Resource Reservations.

1 Introduction

Multimedia systems today support presentations with *continuous media* [1, 18], such as video and audio, as well as synthetic compositions such as slide shows and computer-generated music. We call these presentations *time-based* because they communicate part of their information content through presentation timing. While a query on a database of static data types results in a static view of (hopefully) correct data values, a query for playback of time-based data should result in a presentation with a dynamically changing view. The usefulness of such presentations depends on the accuracy of both the data and the timing. Because digital presentations can only approximate continuous values and timing, the success of playback is a question of *quality* rather than correctness.

Consider the reproduction of NTSC video in a digital multimedia system. The video stream is commonly captured at 640x480 24-bit samples/frame and 30 frames/second, but it is rarely stored or played back at this bandwidth. Instead, lossy compression algorithms such as the MPEG encoding [17] are used to reduce the bandwidth requirements in exchange for some loss in quality. In addition, if the display window does not have the same resolution as the source data then the presentation can only approximate the original data by pixel interpolation.

This observation raises two questions: How accurate must a presentation be, and how can we ensure that a presentation achieves that accuracy? This paper attempts to answer the first question by giving a formal definition of presentation *quality* that measures both accuracy of timing and

*This research is supported by NSF Grants IRI-9223788 and IRI-9117008, and by funds from Tektronix, Inc. and the Oregon Advanced Computing Institute.

the accuracy of output values. This definition of presentation quality can then be used to specify presentation-level quality requirements. The question of how to ensure that quality requirements are met must be answered by a multimedia system. Whenever time-based presentations compete with other applications for resources, some level of guarantees are needed to ensure that resources are not wasted on presentations that provide little value to the user. But without a mechanism for specifying user quality requirements, meaningful guarantees are impossible to request or provide. Section 2 suggests an architecture that derives guarantees for a Quality of Service (QOS) specification as part of an admission test.

QOS specifications for presentation requirements are still a novel concept. Network protocols have been proposed with transport-level QOS specifications that bound delay, minimum throughput and error rates for continuous media communications [12, 16, 18, 31, 32]. More recently, operating systems researchers have argued that *bandwidth reservations* are needed in a real-time operating system to support end-to-end QOS guarantees [3, 10, 14, 20, 23, 25, 33, 35]. Both the network and operating systems QOS goals for bandwidth are typically derived from the type of the data being transmitted, with the assumption that multimedia presentations should deliver as much spatial and temporal resolution as possible. But with current capture, compression, and storage technology, multimedia data types can have resolution that exceeds both the output device capabilities and presentation quality requirements. As the resolution of the data sources increases, users should be able to choose how to sacrifice quality in order to reduce the resource costs of playback.

Many existing multimedia systems make do without QOS-based resource reservations. For example, personal computer systems can successfully play compressed video and audio from CD-ROM, but are able to do so only because the application program has control of all system resources and because the data has been carefully crafted to suit the storage device's throughput and latency [29]. Device independence is possible with adaptive algorithms that adjust the playback quality to the resources available [4, 15, 27, 34]. However, adaptive playback algorithms frequently degrade quality to an unacceptable level when resources overloads occur. A formal definition of quality is needed to specify which presentations are acceptable and what minimal reservations are required to avoid overloads.

This discussion leads to a number of goals for QOS specifications:

- **Model user perception of quality.** The value of a presentation depends on the user's perception of quality, while the cost of a presentation depends on resource usage. Just as modern compression algorithms exploit knowledge of human perception [17, 36], a multimedia system can better optimize playback resources if it knows which optimizations have the least affect on perceived quality.
- **Formal semantics.** Specifications should be unambiguous. A multimedia system should be able to prove that it can satisfy a QOS specification through resource reservations.
- **Support for complex presentations.** Complex presentations can specify synchronization between media streams that originate at independent sources and at different times [19, 21, 29]. QOS specification should apply to any content, and not just a small number of continuous media streams.

This paper defines a framework for specification of presentation QOS. The definitions are intended to be general enough to apply to non-interactive presentations in any multimedia system. The framework considers user interactions for presentation control as interruptions that may require re-computation of the presentation requirements. The next section defines our terminology in terms of an architectural model for multimedia presentations. Sections 3 and 4 elaborate on the specification of *content* and *view* respectively for a presentation. We then define *quality* in Section 5 as a function of a presentation's fidelity to the *content* and *view* specification. Section 6 suggests how a formal QOS specification can be used to optimize resource usage in a presentation. We close with a discussion of related work in Section 7 and our conclusions in Section 8.

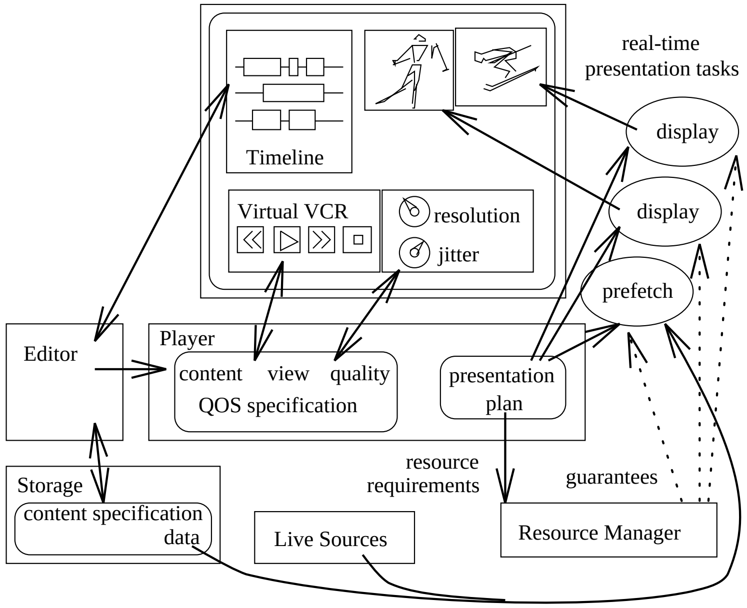

Figure 1: An architecture for editing and viewing multimedia presentations.

2 Architectural Model

In our architectural model, shown in Figure 1, multimedia data comes from live sources or from storage. Digital audio and video data have default content specifications associated with them that specify the sample size and rate for normal playback. A time-based media *editor* may be used to create complex presentations from simple content. A *player* is used to browse and play-back content specified by the editor. A user may control a player's *view* parameters, such as window size and playback rate, as well as *quality* parameters such as spatial and temporal resolution. The combination of content, view, and quality specifications constitute a QOS specification. When a user chooses to begin a presentation, the player needs to verify that a *presentation plan* consisting of real-time tasks will satisfy the *QOS specification*. A presentation plan is feasible if guarantees can be obtained from a *Resource Manager* for the real-time presentation tasks that transport and transform the multimedia data from storage or other data sources to the system outputs.

2.1 Content, View and Quality

This architecture is similar to other research systems that provide QOS guarantees based on an admission test [27]. However, our definition of QOS is novel in that we make strong distinctions between content, view, and quality specifications. A *content* specification defines a set of logical image and audio output values as a function of logical time. A *view* specification maps content onto a set of physical display regions and audio output devices over a real-time interval. *Quality* is a measure of how well an actual presentation matches the ideal presentation of content on a view and a *quality specification* defines a minimum acceptable quality measure. We will refer to quality when we mean the measure, and QOS when we mean the combination of content, view, and quality specifications.

By allowing independent control of content, view and quality, a multimedia system can offer a wider range of services that take advantage of the flexibility of computer platforms. To illustrate these services, consider the presentation of video and audio as described in Figure 2. The first video clip refers to 5 seconds of a digital video file. The video file is named *cam1* because it was captured with the first of two cameras recording the same bicycle racing event. The digital video

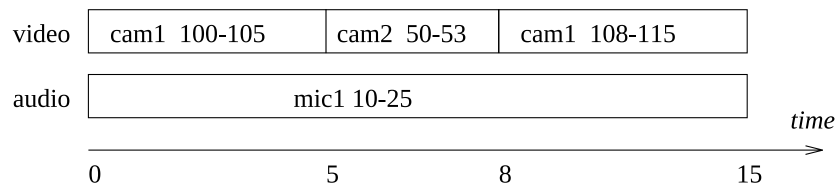

Figure 2: Timeline view of content specification for a presentation of bicycling video with audio.

for cam1 has a resolution of 320x240 pixels. A second video file named *cam2* shows another view of the bicycling event and has a higher resolution of 640x480 pixels. The video presentation cuts from *cam1* to *cam2* for 3 seconds, and then back to *cam1* for the last 7 seconds. The audio clip file *mic1* contains a digital audio sound-track recorded at the same time as the video clips. After selecting this content for presentation, a user should be able to choose view parameters and quality levels independently. For example, if the user chooses a view with a 640x480 pixel display window, but a quality specification that requires only 320x240 pixels of resolution, then the player may be able to avoid generating the full resolution images from cam2. The quality specification allows the user to indirectly control resource usage independent of the content and view selections. The player can optimize resource usage so long as the presentation exceeds the minimum quality specification. Users might also like to specify an upper bound on cost for resource usage, but measuring costs is beyond the scope of this paper.

3 Content Specification

To make the definitions of content, view, and quality as clear as possible, this paper invents a simple framework for specifying multimedia presentations. The Z specification language [28] is used to describe the framework in order to focus attention on the framework's mathematical properties rather than on details of syntax and implementation. The framework supports composition of audio and video data to create complex non-interactive presentations. Other media such as text and still images are supported by converting to a video representation with finite duration.

Our content specifications define a set of logical output channels and the acceptable real-number values for those outputs that may vary continously with time. It is an important feature of this model that the audio and video specifications may have infinite resolution. For example, the visualization of a continuous function whose values can be computed rather than read from storage is limited by the computational resources and the display device, but not by the content specification.

We assume only two basic types: Real numbers and Integers. Digital inputs and outputs will be declared as Integers, but nearly all other quantities will be modeled as Real numbers. Real numbers are used for the specification of logical values to avoid placing an artificial limit on the content resolution. We begin our specification in Z with a declaration of these basic types:

$[R, \mathbb{Z}]$

A *Space* schema specifies intervals for each coordinate dimension and the output value. To make it easier to treat all outputs uniformly, this single schema must contain the maximal set of dimensions for all output types. When used for audio output specifications, we simply ignore the x and y intervals.

Interval
$start : R$
$extent : R$

$$\begin{array}{|l}\hline \textit{Space} \\ t : \textit{Interval} \\ x : \textit{Interval} \\ y : \textit{Interval} \\ z : \textit{Interval} \\ \hline \end{array}$$

A *Content* specification is a recursive construct built from basic audio and video sources. Each *audio*, *video*, *sampledAudio*, and *sampledVideo* construct defines a single logical output channel. More complex content may be specified using *clip*, *transform*, *cat*, *synch*, and *select* constructs. The *LogicalOutput* type is used in the *select* construct to reference a particular logical output. The exact meaning of each of these constructs is described below.

$$\begin{array}{lll} \textit{LogicalOutput} & ::= & \textit{aLog} \; \langle\!\langle \mathbb{Z} \rangle\!\rangle \mid \textit{vLog} \; \langle\!\langle \mathbb{Z} \rangle\!\rangle \\ \textit{Content} & ::= & \textit{audio} \langle\!\langle \textit{Space} \times (R \longrightarrow R) \rangle\!\rangle \\ & | & \textit{video} \langle\!\langle \textit{Space} \times (R \longrightarrow R \longrightarrow R \longrightarrow R) \rangle\!\rangle \\ & | & \textit{sampledAudio} \langle\!\langle \textit{Space} \times (\mathbb{Z} \longrightarrow R) \rangle\!\rangle \\ & | & \textit{sampledVideo} \langle\!\langle \textit{Space} \times (\mathbb{Z} \longrightarrow \mathbb{Z} \longrightarrow \mathbb{Z} \longrightarrow R) \rangle\!\rangle \\ & | & \textit{clip} \langle\!\langle \textit{Space} \times \textit{Content} \rangle\!\rangle \\ & | & \textit{transform} \langle\!\langle \textit{Space} \times \textit{Content} \rangle\!\rangle \\ & | & \textit{cat} \langle\!\langle \mathrm{seq} \;\; \textit{Content} \rangle\!\rangle \\ & | & \textit{synch} \langle\!\langle \mathrm{seq} \;\; \textit{Content} \rangle\!\rangle \\ & | & \textit{select} \langle\!\langle \textit{LogicalOutput} \times \textit{Content} \rangle\!\rangle \end{array}$$

Figure 3 illustrates a content specification for the example presentation from Figure 2. We'll describe this specification from the bottom up, beginning with the two *sampledVideo* specifications and the one *sampledAudio* specification that form the leaves of the tree. The first video specification declares that the file *cam1* contains 8-bit samples in 3450 frames and each frame has 320x240 pixels. The second video file named *cam2* has only 1590 frames, but each frame has 640x480 pixels. The audio file *mic1* has 100 seconds worth of samples at 8000 8-bit samples/second. Both videos are scaled in time to play at 30 frames/second and their z ranges are normalized by the *transform* specifications. The first video is also scaled by a factor of 2 to match the dimensions of the second video and is offset by -100 seconds so that the clip can begin at logical time zero. The audio is normalized and scaled in time to play at 8000 samples/second. The video presentation is assembled by concatenating a clip of seconds 0-5 from the first transformed video with seconds 50-53 from the second, followed by the clip of seconds 8-15 from the first again. The result is then synchronized with a clip of seconds 0-15 from the transformed audio.

The *transform*, *clip*, *cat*, *synch*, and *select* specifications support stretching and shrinking, cut, paste, and synchronization of logical outputs. Although other features are desirable, such as the ability to mix several logical outputs together, the constructs described are sufficient for editing useful time-based multimedia presentations and for illustrating the meaning of view and quality specifications in the next sections.

3.1 Meaning of Content Specifications

The meaning of a content specification is defined by a set of allowed logical output values for every point of the logical output space. A few more declarations make it easier to define this logical meaning. We first introduce the notation $r \in_I i$ to express the constraint that a real number r is within the interval i. The Z notation for declaring the relation $\in_I$ is:

$$\begin{array}{|l} \_ \in_I \_ : R \longleftrightarrow \textit{Interval} \\ \hline r \in_I i = (i.\textit{start} \leq r) \wedge (r < i.\textit{start} + i.\textit{extent}) \end{array}$$

We also declare two functions *tr* and *utr* that respectively transform and untransform a real

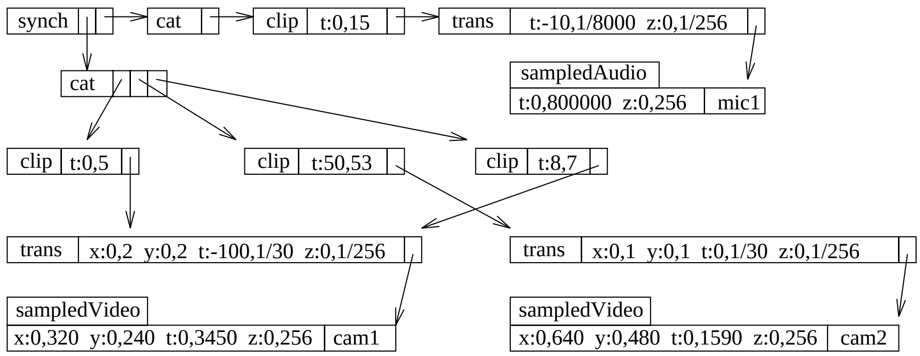

Figure 3: Content specification in normal-form for example presentation.

number r by a scale factor $i.extent$ and an offset $i.start$. For example, if i is an *Interval* with $i.start = 3.0$ and $i.extent = 1.5$ then $tr\ 6.5\ i = 12.75$.

$$tr, utr : R \longrightarrow Interval \longrightarrow R$$

$$tr\ \ r\ \ i = r * i.extent + i.start$$

$$utr\ \ r\ \ i = (r - i.start)/i.extent$$

Content specifications constrain logical output values *only* during explicit time intervals. For example, the content specification in Figure 3 allows any output values before logical time 0 and after logical time 15. The functions *start*, *end*, and *duration* are used to reference the logical time interval over which output values are constrained by a content specification. The logical start of a content specification is the minimum time t at which *some* output value is *not* acceptable! The logical end is the minimum time t such that no output value is constrained for times greater than or equal to t. The function *aLogs* returns the integral number of logical *audio* outputs that are constrained by a content specification and *vLogs* returns the number of logical *video* outputs.

$$start, end, duration : Content \longrightarrow R$$
$$aLogs, vLogs : Content \longrightarrow \mathbb{Z}$$

$$start\ \ c = min\ \{\ t : R\ \ |\ \ \neg\ (\forall\, l : LogicalOutput;\ x, y, z : R \bullet (l, x, y, t, z) \in logical\ \ c)\ \}$$

$$end\ \ c = min\ \{\ t : R\ \ | \\ \quad \forall\, t' : R \bullet (t \leq t') \Rightarrow (\forall\, l : LogicalOutput;\ x, y, z : R \bullet (l, x, y, t', z) \in logical\ \ c)\ \ \}$$

$$duration\ \ c = end\ \ c - start\ \ c$$

$$aLogs(c) = max\ \{\ n : \mathbb{Z}\ |\ \ \neg\ (\forall\, x, y, t, z : R \bullet (aLog(n), x, y, t, z) \in logical\ \ c)\ \}$$

$$vLogs(c) = max\ \{\ n : \mathbb{Z}\ |\ \ \neg\ (\forall\, x, y, t, z : R \bullet (vLog(n), x, y, t, z) \in logical\ \ c)\ \}$$

The meaning of each of the content constructs is captured by the following definition of a function for logical content. For a given content specification, the *logical* function returns a relation between a point in the logical output space and the acceptable output values for that point. We read the expression $(l, x, y, t, z) \in logical\ c$ as: the content specification c specifies that logical output l, at point (x, y) and time t may have value z. Note that specifications reduce the set of acceptable values and where nothing is specified, all values are acceptable.

$$LogicalValue == LogicalOutput \times R \times R \times R \times R$$

$$
\begin{array}{l}
logical : Content \longrightarrow \mathbb{P}\, LogicalValue \\
\hline
logical(audio(s,f)) = \{\, l : LogicalOutput;\ x,y,t,z : R \mid \\
\quad (l = aLog\ 1) \land (t \in_I s.t) \Rightarrow z = f\ t \bullet (l,x,y,t,z) \,\} \\
logical(video(s,f)) = \{\, l : LogicalOutput;\ x,y,t,z : R \mid \\
\quad (l = vLog\ 1) \land (x \in_I s.x) \land (y \in_I s.y) \land (t \in_I s.t) \Rightarrow z = f\ t\ y\ x \bullet (l,x,y,t,z) \,\} \\
logical(sampledAudio(s,f)) = \{\, l : LogicalOutput;\ x,y,t,z : R \mid \\
\quad (l = aLog\ 1) \land (t \in_I s.t) \land z = f\ \lfloor t \rfloor \bullet (l,x,y,t,z) \,\} \\
logical(sampledVideo(s,f)) = \{\, l : LogicalOutput;\ x,y,t,z : R \mid \\
\quad (l = vLog\ 1) \land (x \in_I s.x) \land (y \in_I s.y) \land (t \in_I s.t) \Rightarrow z = f\ \lfloor t \rfloor\ \lfloor x \rfloor\ \lfloor y \rfloor \bullet (l,x,y,t,z) \,\} \\
logical(clip(s,c)) = \{\, l : LogicalOutput;\ x,y,t,z : R \mid \\
\quad (x \in_I s.x) \land (y \in_I s.y) \land (t \in_I s.t) \land (z \in_I s.z) \Rightarrow \\
\qquad (l,x,y,t,z) \in logical\ c \bullet (l,x,y,t,z) \,\} \\
logical(transform(s,c)) = \{\, l : LogicalOutput;\ x,y,t,z : R \mid \\
\quad (l,x,y,t,z) \in logical\ c \bullet (l, tr\ x\ s.x, tr\ y\ s.y, tr\ t\ s.t, tr\ z\ s.z) \,\} \\
logical(cat(\langle\rangle)) = \{\, l : LogicalOutput;\ x,y,t,z : R \bullet (l,x,y,t,z) \,\} \\
logical(cat(q)) = \\
\quad logical(head(q)) \\
\quad \cap \{\, l : LogicalOutput;\ x,y,t,z : R \mid \\
\qquad (l,x,y,t,z) \in logical(cat(tail(q))) \bullet (l,x,y,t + end(head(q)) - start(cat(tail(q))), z) \,\} \\
logical(synch(\langle\rangle)) = \{\, l : LogicalOutput;\ x,y,t,z : R \bullet (l,x,y,t,z) \,\} \\
logical(synch(q)) = \\
\quad \{\, n : \mathbb{Z};\ x,y,t,z : R \mid (aLog\ n,x,y,t,z) \in logical\ head(q) \bullet \\
\qquad (aLog(n + (aLogs(synch(tail(q))))), x,y,t,z) \,\} \\
\quad \cap \{\, n : \mathbb{Z};\ x,y,t,z : R \mid (vLog\ n,x,y,t,z) \in logical\ head(q) \bullet \\
\qquad (vLog(n + (vLogs(synch(tail(q))))), x,y,t,z) \,\} \\
\quad \cap\ logical(synch\ tail(q)) \\
logical(select(aLog(n),c)) = \{\, x,y,t,z : R \mid (aLog(n),x,y,t,z) \in logical\ c \bullet (aLog(1),x,y,t,z) \,\} \\
logical(select(vLog(n),c)) = \{\, x,y,t,z : R \mid (vLog(n),x,y,t,z) \in logical\ c \bullet (vLog(1),x,y,t,z) \,\}
\end{array}
$$

Some explanation is needed regarding audio and video source functions. An audio source is modelled as a function from a real time coordinate to a signal value. For example, a sine function could be given as an audio source without specifying a limit on the resolution of the signal. As described in the following sections, the resolution of a presentation is limited only by an actual implementation on digital outputs. A video source is also modelled as a continuous function of time, but it requires additional arguments for the x and y screen coordinates. The domain and range for the functions are specified with the *Space* argument. Sampled audio and video sources are modelled as functions of integer coordinates. For simplicity, this definition supports only monochrome video, but the same approach can be generalized to specify a tuple of values at each point for color.

The first predicate for $logical(audio(s,f))$ says that if l is the logical output $aLog\ 1$ and t is within the interval $s.t$ then the only acceptable value for z is the function $f(t)$. Otherwise, any values are acceptable for z. Note that the interval $s.z$ may indicate the intended range of source values, but there is no need to enforce this range when defining the logical content. The predicate for $logical(video(s,f))$ expresses a similar constraint for the logical output $vLog1$.

For sampled audio and video, the logical coordinates are rounded down to the nearest integer. Consequently, the number of samples (frames) is given by $\lfloor s.t.extent \rfloor$ and the pixel dimensions for video frames is $\lfloor s.x.extent \rfloor * \lfloor s.y.extent \rfloor$. This information about sample resolution is needed only for accessing the source functions and is not carried explicitly in the definition of logical content.

A $clip(s, c)$ construct specifies that for all logical outputs, points within the *Space* s are constrained to have the same values as specified by c. All points not in s are effectively "clipped" out and may have any value. A $transform(s, c)$ construct specifies a linear transformation of points in the content specified by c. For example, if $start\ c = 0$, $duration\ c = 60$, $s.t.start = 10$, and $s.t.extent = 2$, then $start(transform(s, c)) = 10$ and $duration(transform(s, c)) = 120$. The transformation construct $transform(s, c)$ with all start fields in s equal to zero and all extent fields in s equal to one is the identity transformation and has no effect.

A temporal sequence of content can be specified with a $cat(q)$ construct. The content for a member of the sequence q is logically shifted in time to start just as the previous content in the sequence ends. The $synch(q)$ construct specifies that a set of logical outputs all reference the same time scale. If the content specifications c_n and c_m specify n and m logical outputs respectively, then $synch(\langle c_n, c_m \rangle)$ specifies $m + n$ logical outputs.

The $select(l, c)$ construct offers a way to reference only the content of a single logical output within a complex specification. Where the *synch* construct aggregates multiple logical outputs into a single content specification, $select(l, c)$ specifies only a single logical output with the same content as c specifies for logical output l. For any n, the logical output defined by $select(l, c)$ is $(aLog\ 1)$ if $l = (aLog\ n)$ and $(vLog\ 1)$ if $l = (vLog\ n)$. If a content construct does not specify the logical output l then $select(l, c)$ is the null specification; that is, all values are permissible on all outputs.

It is worth noting that no matter how a content specification is composed, its logical content may be equivalently specified by a content specification with the normal-form shown in Figure 3. In normal-form, every specification is a tree with a *synch* construct at the root. The *synch* construct specifies a sequence of *cat* constructs. Each *cat* construct specifies a single logical output with a sequence of *clip* constructs. Each *clip* specifies a portion of a *transform* construct and each *transform* construct defines the logical dimensions of a basic media source. A basic media source must be either an *audio*, *video*, *sampledAudio*, or *sampledVideo* construct.

4 View Specification

The logical outputs of a content specification have both temporal and spatial proportions, but they have no physical size or real duration. A *View* specification allocates physical devices for logical outputs and maps logical time to a real-time clock. While the physical devices may present an upper bound on spatial and temporal resolution, the view does not specify presentation quality. Figure 4 shows a view specification that allocates an unusually small 8x6 pixel window on a monochrome (black and white) display for the bicycling video presentation. Although the output device clearly limits the quality of the presentation, the view does not specify how the content is to be represented on the display. It is the presentation plan that must choose how to resample the source and how to represent gray scale information. The combination of content and view specifications serve as a device-independent specification of a perfect quality presentation. The idea of an ideal presentation is formally defined below. In the next section, we define less-than-perfect quality based on the difference between this ideal presentation and actual presentation outputs.

Since we are not interested in the details of the physical device I/O, we simply assume that there is a set of audio output devices *AudioDev* and video output devices *VideoDev*. A *Device* is either one of the audio devices or one of the video devices.

$[AudioDev, VideoDev]$

$Device == AudioDev \cup VideoDev$

The logical dimensions in a content specification are generally not the same as the physical dimensions of the view. The *Output* schema declares a field *tr* that defines the transformation from logical to view output dimensions and a field *clip* that defines clipping bounds for view outputs. In Figure 4, the *Output* specification for $vLog\ 1$ transforms the 640x480 logical image size to 8x6 and then offsets the image 2 pixels in x and 1 in y. The z values are transformed from the logical

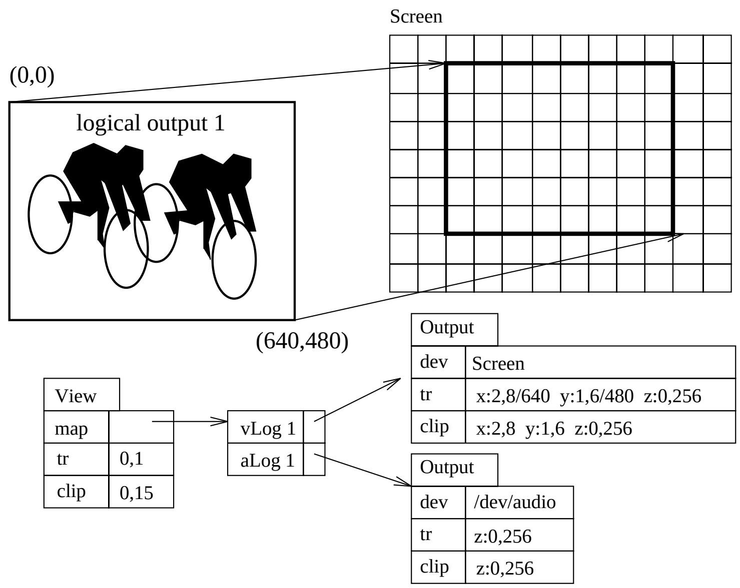

Figure 4: Example of a view that allocates an 8x6 pixel window on a display device for presentation of the bicycling video.

range of $[0, 1)$ to the view range of $[0, 256)$. The clipping bounds for both audio and video match the full range of the transformed content. Note that the time fields are ignored, because the temporal transformation and clipping for all outputs is given in the *View* specification. This assymetry is necessary to preserve the content synchronization while allowing flexibility in the display of multiple logical outputs.

Output
$dev : Device$
$tr, clip : Space$

A *View* specifies a partial function *map* that assigns a subset of the logical outputs to physical output specifications. Logical content that is to be presented must be mapped to physical outputs of the appropriate type. Logical outputs that are not in the domain of the map function are ignored. The *tr* field is used to transform logical time in a content specification to a real-time clock. The *clip* field specifies the real-time start and duration of the presentation.

View
$map : LogicalOutput \nrightarrow Output$
$tr : Interval$
$clip : Interval$

$(aLog\ n \in \mathrm{dom}\, map) \Rightarrow (\exists\, d : AudioDev \bullet\ (map(aLog\ n)).dev = d)$

$(vLog\ n \in \mathrm{dom}\, map) \Rightarrow (\exists\, d : VideoDev \bullet\ (map(vLog\ n)).dev = d)$

A view specification together with a content specification defines an *ideal* presentation, where the output devices are assumed to have infinite resolution. This assumption is necessary for the device-independent definition of quality described in the next section.

$DeviceValue == Device \times R \times R \times R \times R$

We define a function *ideal c v* that returns the relation between devices and the values specified by a *Content* specification c and a *View* specification v. The relation *ideal c v* contains all points (d, x, y, t, z), where the view maps a logical output l to a device d and x, y, and t are within the clipping bounds for d, only if the corresponding logical value is allowed by the content specification c. The corresponding logical point is computed by substituting l for p and "un-transforming" x, y, t, and z back to logical space.

$$ideal : Content \longrightarrow View \longrightarrow \mathbb{P}\, DeviceValue$$

$$\begin{array}{l}
ideal\ c\ v = \{\, d : Device;\ x, y, t, z : R \bullet \\
\quad (\exists\, l : LogicalOutput;\ p : Output \bullet \\
\qquad ((l \in \operatorname{dom} v.map) \wedge (v.map(l) = p) \wedge (p.dev = d) \\
\qquad \wedge\ (d \in AudioDev) \wedge (t \in_I v.clip)) \Rightarrow \\
\qquad\quad (\exists\, x', y' : R \bullet (l, x', y', utr\ t\ v.tr, utr\ z\ p.tr.z) \in logical\ c)) \\
\quad \wedge\ (\exists\, l : LogicalOutput;\ p : Output \bullet \\
\qquad ((l \in \operatorname{dom} v.map) \wedge (v.map(l) = p) \wedge (p.dev = d) \\
\qquad \wedge\ (d \in VideoDev) \wedge (t \in_I v.clip) \wedge (x \in_I p.clip.x) \wedge (y \in_I p.clip.y)) \Rightarrow \\
\qquad\quad ((l, utr\ x\ p.tr.x, utr\ y\ p.tr.y, utr\ t\ v.tr, utr\ z\ p.tr.z) \in logical\ c)) \\
\quad \bullet (d, x, y, t, z) \,\}
\end{array}$$

The implementation of a presentation plan uniquely determines the value for every device at every point and time. The schema *Presentation* models the implementation with separate functions for audio and video device types. A presentation on a digital audio device is a function from a discrete clock value for time to an integer output value. For a video output, it is a function from a discrete clock value and integer x and y coordinates to an integer output value.

Presentation

$$\begin{array}{l}
aVal : AudioDev \longrightarrow \mathbb{Z} \longrightarrow \mathbb{Z} \\
vVal : VideoDev \longrightarrow \mathbb{Z} \longrightarrow \mathbb{Z} \longrightarrow \mathbb{Z} \longrightarrow \mathbb{Z}
\end{array}$$

We define a function *actual* that takes a particular presentation and returns a relation representing these output values. The relation *actual P* contains a point (d, x, y, t, z) only if z is the value of the device d and pixel (x, y) while the clock value is t as defined by P. We are assuming that we can observe only one output value per clock tick and that the output value is constant over the duration of a clock cycle. Because the relation *actual P* and the relation *ideal c v* have the same type, it is easy to define a mapping between them for any presentation P, content specification c, and view v.

$$actual : Presentation \longrightarrow \mathbb{P}\, DeviceValue$$

$$\begin{array}{l}
actual\ P = \\
\quad \{\, d : Device;\ x, y, t : R \mid \\
\qquad d \in AudioDev \bullet (d, x, y, t, P.aVal\ d\ \lfloor t \rfloor) \,\} \\
\quad \cup \{\, d : Device;\ x, y, t : R \mid \\
\qquad d \in VideoDev \bullet (d, x, y, t, P.vVal\ d\ \lfloor x \rfloor\ \lfloor y \rfloor\ \lfloor t \rfloor) \,\}
\end{array}$$

5 Quality Specification

We define the *quality* of a presentation to be the ratio of the *worth* of the actual presentation to the worth of the ideal presentation. Although worth may be subjective, we believe the ratio can be usefully modelled with a few assumptions:

1. User perception of presentation quality can be modelled by a continuous function of real-time and device coordinates.
2. The quality of a presentation that meets the specification equals one.

3. The quality of a presentation that differs from the specification depends only on user perception of the difference.

4. User perception of the difference between a presentation and the specification is based on a mapping of points in the actual presentation to points in the ideal presentation.

With these assumptions, quality is independent of data representations and transport mechanisms. In particular, our definition of quality is not based on the data throughput required for a presentation, but instead can be used to determine throughput requirements as shown in the next section. In this section, we provide a model for computing quality and define quality specifications in terms of this model.

The declaration for an *ErrorInterpretation* below is the most important part of our QOS specification because it defines an *error model* for measuring presentation quality. An *error model* is a set of functions that describe the number of ways in which an actual presentation may be different from an ideal presentation. We refer to these functions as *error components*.

Figure 5 gives a simple example of the need for an adequate error model. In the first graph, the difference between the actual and ideal curves at time t gives a fair measure of the perceived error. In the second graph the same measurement at time t gives a very large error measurement, even though most users will recognize that the signal was simply shifted to start at time $t2$. The same error can be explained in several ways, but the addition of more error components to an error model allows greater opportunities to distinguish acceptable errors from unacceptable ones. We can more accurately model the way users perceive the error in Figure 5 if we include both value error and time-shift error components in our model. But an error model with only these two components is still inadequate for the common errors that occur in multimedia presentations.

Our error model below proposes a set of error components that correspond to well known quality parameters. This set of error components both extends the quality parameters proposed by others and gives them a formal definition. Our calculation of presentation quality can be improved by extending or customizing the error model.

Several type declarations and functions simplify the definition of our error model and quality constraints. An *xytFun* is just an abbreviation for a function that takes three real numbers for x, y and t coordinates and returns a real number.

$$xytFun == R \longrightarrow R \longrightarrow R \longrightarrow R$$

The *Error* data type provides names for the error components that our error model associates with each output. The motivation for this set of error components is described in Section 5.1.

$$Comp ::= err \mid shift \mid rate \mid jitter \mid res$$
$$Error ::= X\langle\langle Comp\rangle\rangle \mid Y\langle\langle Comp\rangle\rangle \mid T\langle\langle Comp\rangle\rangle \mid Z\langle\langle Comp\rangle\rangle$$

Our error model also includes a function for the synchronization error between each pair of outputs. The reasons for choosing this particular set of error components is discussed at the end of this section. In particular though, it is useful to consider spatial and temporal resolution in order to correctly model user perception of output values. The function $localAvg\ (xres, yres, tres)\ f$ computes an *xytFun* that is the average value of the function f over a small local area defined by $xres$, $yres$, and $tres$. Because audio outputs do not vary in x or y, $localAvg\ (X\ \ res, Y\ \ res, T\ \ res)\ f$ is independent of the values of $X\ \ res$ and $Y\ \ res$ in that case and is therefore well defined even when $X\ \ res$ and $Y\ \ res$ are not specified.

$$
\begin{array}{l}
localAvg : (xytFun \times xytFun \times xytFun) \longrightarrow xytFun \longrightarrow xytFun \\
\hline
localAvg\ (xres, yres, tres)\ f\ x\ y\ t = \\
\quad (\textbf{let}\ x_r = xres\ x\ y\ t;\ x_1 = x - (x_r/2);\ x_2 = x + (x_r/2); \\
\qquad y_r = yres\ x\ y\ t;\ y_1 = y - (y_r/2);\ y_2 = y + (y_r/2); \\
\qquad t_r = tres\ x\ y\ t;\ t_1 = t - (t_r/2);\ t_2 = t + (t_r/2) \bullet \\
\qquad\quad \frac{1}{x_r * y_r * t_r} \int_{t_1}^{t_2} \int_{y_1}^{y_2} \int_{x_1}^{x_2} (f\ x'\ y'\ t')\ dx'\ dy'\ dt')
\end{array}
$$

ErrorInterpretation

$$
\begin{array}{l}
c : Content \\
v : View \\
P : Presentation \\
error : Output \longrightarrow Error \longrightarrow xytFun \\
synch : Output \longrightarrow Output \longrightarrow xytFun \\
\hline
\forall\, p : Output \bullet \textbf{let}\ i == error\ p \bullet \\
\quad \exists\, z_{ideal}, z_{actual} : xytFun \bullet \\
\qquad (\forall\, x, y, t : R \bullet (p, x, y, t, z_{ideal}\ x\ y\ t) \in ideal\ c\ v) \\
\qquad \wedge (\forall\, x, y, t : R \bullet (p, x, y, t, z_{actual}\ x\ y\ t) \in actual\ P) \\
\qquad \wedge\ i(Z\ err) = (\lambda\ x, y, t : R \bullet \\
\qquad\quad z_{actual}\ x\ y\ t - \\
\qquad\quad z_{ideal}(x + i(X\ err)\ x\ y\ t)(y + i(Y\ err)\ x\ y\ t)(t + i(T\ err)\ x\ y\ t)) \\
\qquad \wedge\ i(X\ err) = i(X\ shift) + i(X\ jitter) \\
\qquad \wedge\ i(Y\ err) = i(Y\ shift) + i(Y\ jitter) \\
\qquad \wedge\ i(T\ err) = i(T\ shift) + i(T\ jitter) \\
\qquad \wedge\ i(X\ rate) = \partial i(X\ shift)/\partial x \\
\qquad \wedge\ i(Y\ rate) = \partial i(Y\ shift)/\partial y \\
\qquad \wedge\ i(T\ rate) = \partial i(T\ shift)/\partial t \\
\qquad \wedge\ localAvg\ (i(X\ res), i(Y\ res), i(T\ res))\ (i(Z\ err)) = \\
\qquad\quad (\textbf{let}\ perceivedErr = i(Z\ shift) + ((1 + i(Z\ rate)) * z_{ideal}) + i(Z\ jitter) \bullet \\
\qquad\qquad localAvg\ (i(X\ res), i(Y\ res), i(T\ res))\ perceivedErr) \\
\\
\forall\, p, q : Output \bullet \\
\quad synch\ p\ q = (error\ p\ (T\ shift)) - (error\ q\ (T\ shift))
\end{array}
$$

The error model in this declaration defines a set of error components for each output through the *error* function as well as an error component for each pair of outputs defined by the function *synch*. The predicate for an *ErrorInterpretation* is like a differential equation in that it does not have a unique solution for the error component functions. Instead, we observe that error measurement is inherently subjective because the outputs do not carry meta-information about the intended relationship with the specification. An *ErrorInterpretation* merely defines one subjective mapping and a set of error components that are consistent with each other and the mapping. Figure 5 illustrates the point with two different interpretations for an audio presentation.

We declare a *quality* specification to be a schema that gives the minimum acceptable level of quality and also provides values for calibrating the affect of each error component on presentation quality.

Quality

$$
\begin{array}{l}
min : R \\
calib : Output \longrightarrow Error \longrightarrow R \\
calibSynch : Output \longrightarrow Output \longrightarrow R \\
\hline
(0 \leq min) \wedge (min < 1)
\end{array}
$$

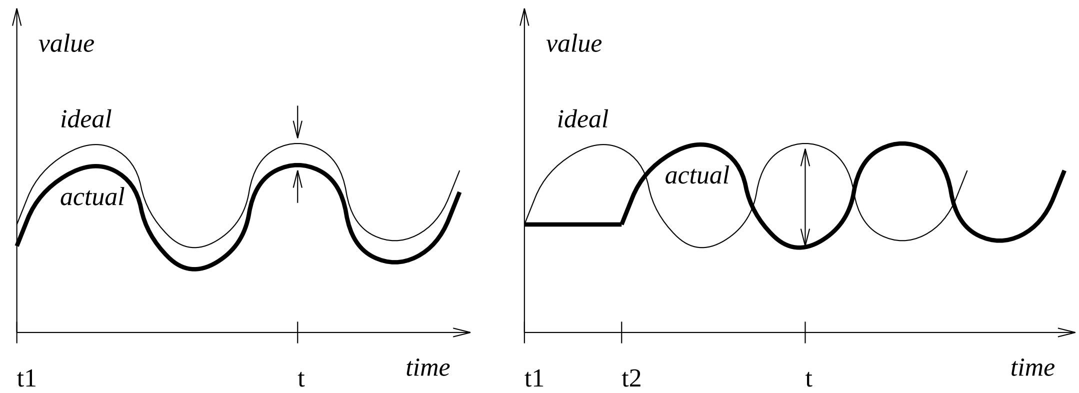

Figure 5: Presentation error may be attributed to value error alone as illustrated or to some combination of timing and value errors.

The meaning of the quality schema in conjunction with a content and view specification is given by the following schema for a *QOS* specification:

QOS

$c : Content$
$v : View$
$q : Quality$
$P : Presentation$

$\exists\, i : ErrorInterpretation \bullet i.c = c \wedge i.v = v \wedge i.P = P \wedge$
$\quad \forall\, p \in \operatorname{ran} v.map;\ x, y, t : R \bullet$
$$q.min \le \prod_{m \in Error} exp(- \mid \frac{i.error\ p\ m\ x\ y\ t}{q.calib\ p\ m} \mid) * \prod_{p' \in \operatorname{ran} v.map} exp(- \mid \frac{i.synch\ p\ p'\ x\ y\ t}{q.calibSynch\ p\ p'} \mid)$$

This schema consists of *Content*, *View*, and *Quality* specifications that constrain a presentation P. The *QOS* specification is satisfied only if an *ErrorInterpretation* exists for c v and P such that, at all times and all points on every output, the quality of the presentation must be greater than or equal to $q.min$. We compute quality with an exponential decay function that depends on the absolute value of error components. This model has the following properties:

- quality is one when all error components are zero.
- quality is monotonically decreasing with increasing absolute value of any error component.
- quality approaches zero as all error components approach infinity.

To calibrate this quality function to approximate user preferences, we can adjust the values returned by the *calib* and *synchCalib* functions in the *quality* specification. We call these values *critical error values.* For every error component in our error model, there is a corresponding critical error value in the *quality* specification. When an error component equals the corresponding critical error value the quality is at most e^{-1} or approximately 0.37. Consequently, we must choose these critical error values to correspond to decidedly poor quality. Figure 6 shows critical error values for the example in the next section. A quality specification q can use these values for its *calib* and *synchCalib* functions. For example, for all video outputs p, *q.calib p (T jitter)* is 0.1 seconds. These numbers are intended to correspond to noticeably poor quality. The units for temporal *shift*, *jitter*, *res*, and *synch* are in seconds. Measurements for x and y *shift*, *jitter*, and *res* components are relative to $v.clip.x.extent$ and $v.clip.y.extent$. respectively for a *View* v. Measurements for z *shift*

	shift	rate	jitter	res	synch
VideoDev X	0.1	0.1	0.02	0.02	0.2
VideoDev Y	0.1	0.1	0.02	0.02	
VideoDev T	15	0.5	0.1	0.03	
VideoDev Z	0.05	0.1	0.05		
AudioDev T	15	0.5	0.001	0.0002	
AudioDev Z	0.01	0.1	0.05		

Figure 6: Example critical error values. The "err" component does not appear in this table because it is equivalent to the sum of the "shift" and "jitter" components.

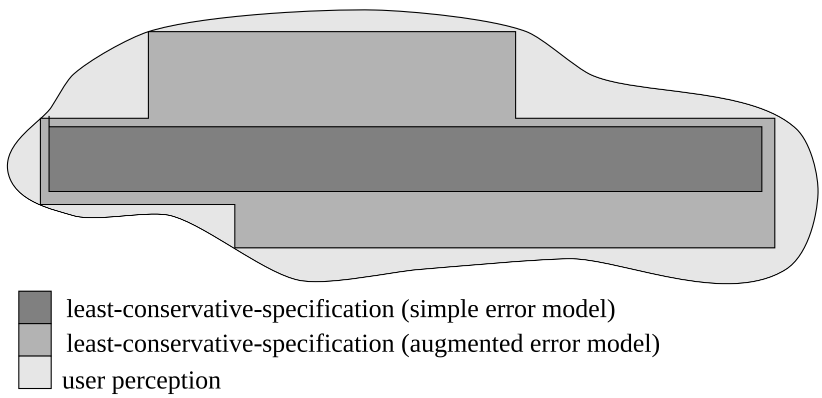

Figure 7: Relationship between presentations accepted by least-conservative-specifications and those accepted by user perception.

and *jitter* are also made relative to $v.clip.z.extent$. All *rate* error components are measured in units of *shift* per second.

This definition for QOS specification is very strict in that quality must exceed the minimum *everywhere* during a presentation. It would be nice to extend the specification semantics to allow a presentation to occasionally drop below this minimum quality, but this extension is left for future work.

5.1 Justifying the error model

The choice of error components in our error model is intended to provide a useful model of human perception. Ideally, a presentation QOS specification should accept all presentations that humans accept and reject only those that humans reject. A *conservative* specification is one that never accepts a presentation that humans would reject and a *least conservative* specification is a conservative specification that accepts the largest set of presentations. We can show that a least conservative QOS specification for a minimal error model needlessly rejects presentations that we find acceptable. Error components are added to the error model to increase the space of presentations accepted by a least conservative specification as suggested in Figure 7.

Consider the minimal error model that includes a function for error in the Z dimension for every output, but that assumes error in X, Y, and T dimensions are zero. This simple model is illustrated in Figure 5 where the error function returns the minimum difference between the actual output value z and an ideal value for each output, p, at the same point and time.

This minimal error model is the smallest error model that can map from *actual* P to *ideal* c v for any P, c, and v. We say that an error model is *complete* if we can specify arbitrarily high quality (as judged by humans) by requiring that all error components in the error model are sufficiently close to zero. A complete error model is essential for conservative QOS specifications. The minimal error model is complete because the presentation becomes indistinguishable from the ideal as the Z error component goes to zero everywhere.

Unfortunately, the minimal error model does not yield error values that correspond well to human perception. The second case in Figure 5 shows that a simple startup delay produces large error measurements. A person judging the quality of a presentation recognizes a delay in starting the presentation, but then sees a good match after compensating for the delay. If human ears are annoyed by noise with an amplitude e_a then a conservative QOS specification must constrain error in Z to be less than e_a. This constraint limits the set of acceptable presentations to a narrow band around the ideal presentation. But the human listener will accept the much larger set of presentations that are merely delayed in starting by up to $\pm 1/10$ second. By adding a constant time shift error to the error model, we can accept this much larger set of presentations with a QOS specification that is still conservative.

Our error model adds many error components to achieve a better match between the least conservative QOS specification and human perception. The *shift* component for time is intended to express the amount by which a presentation is seen to be behind schedule. The *shift* error need not be constant in our model, but may increase or decrease with time. We add the *synch* error component because shift errors are not likely to be noticed except as part of a synchronization error between outputs. Humans are sensitive to the presentation rate, so the error model includes *rate* error which we define as the rate of change in the shift error with respect to time. The *rate* error is zero while the *shift* error is constant, but increases in magnitude when the presentation speeds up or slows down. The addition of *jitter* to the error model allows us to isolate high frequency noise in the timing from the *shift* and *rate* components. For example, logical time for a video might be accurately perceived as being stopped between frames and then advancing rapidly as the next frame is presented. Rather than reflecting this rate fluctuation in the *rate* error component, the *jitter* error component accounts for these small timing errors. As discussed below, the error model does not need to specify how much of the timing error is due to *jitter* and how much to *shift*.

The *shift*, *rate*, and *jitter* error components are defined similarly for X and Y dimensions since video presentations can suffer from displacement, scaling and small distortions that are analogous to the temporal error components.

Even after accounting for temporal and spatial errors, the difference between an actual presentation value and the corresponding ideal value at an infinitesimal point is not particularly meaningful. The problem is that humans don't perceive independent values at infinitesimal points, but instead integrate over small display areas and time intervals. This fact is routinely exploited by graphics algorithms that use dithering. For example, a black and white display can represent a 50% gray tone by a pattern with every other pixel turned on. Dithering trades off spatial resolution for more accurate average values. Let $X\ res$ be the width of the smallest resolvable vertical stripe in a presentation. We define $Y\ res$ and $T\ res$ similarly. Then the interesting measure of value error is the difference in average value over a region with dimensions $X\ res * Y\ res * T\ res$. This separates value errors into what we perceive as resolution loss and actual "wrong" values. Our error model includes Z *shift*, Z *rate*, and Z *jitter* error components to model offset, scale, and noise errors respectively. All three are related to the value of $Z\ err$ averaged over a region defined by the resolution error components.

The determination of error component functions is inherently ambiguous because there is no information in an output signal about the *intended* correspondence with a specification. Each user perceives error in a presentation subjectively, and may assess the error differently. For a given presentation and its specification there are an infinite number of interpretations that will satisfy our error model, each with a different affect on presentation quality. What matters is that an interpretation exists that has acceptable errors. We assume that users are good at recognizing the intended presentation content and that they therefore perceive the interpretation with errors that

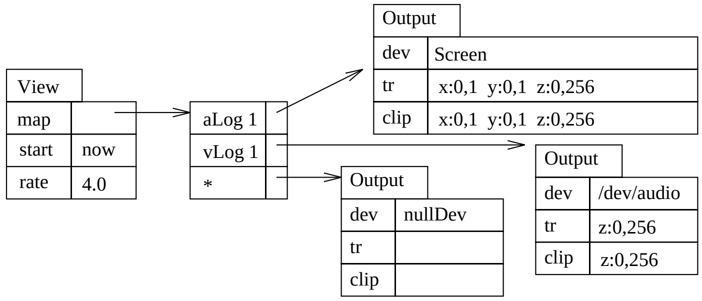

Figure 8: View specification for playback of bicycling video at four times normal rate.

are the most acceptable.

We say that an error model is *sound* if, for any trio of *Content*, *View*, and *Presentation*, a set of error component functions exist that satisfy the definition of the error model. The error model proposed in this section is both sound and complete and also gives formal definitions for shift, rate, jitter, resolution and synchronization errors that are a superset of the QOS parameters proposed by other researchers [7, 14, 20, 32]. The utility of a particular error model depends in part on how well it models human perception of errors that affect quality. Further work is needed to evaluate the utility of this particular error model.

6 Using Quality Specifications for Resource Reservation

A multimedia player can generally meet a QOS specification with fewer resources than are needed for a maximal quality presentation. Consider the bicycling video of Figure 3 and a new view specification shown in Figure 8. Let the quality specification q have the critical error values from Figure 6 and the value 0.75 for $q.min$. The view represents a user request to play the presentation at 4 times the normal rate. The resulting ideal specification then calls for 120 frames/second of video. However, the quality specification only requires that quality exceed 0.75. If all aspects of the presentation were perfect except for video jitter, the quality specification would admit a presentation with jitter less than or equal to 0.029 seconds, which allows the playback algorithm to drop more than five out of six frames. This result follows from the predicate in the *QOS* schema.

Let p_v be the video output and i be an interpretation that finds all error components to be zero except for $i.error\ p_v\ (T\ jitter)$. Since the exponential functions are equal to one when error is zero, we get:

$$\forall\, x, y, t : R \bullet 0.75 \leq exp(- \mid \frac{i.error\ p_v\ (T\ jitter)\ x\ y\ t}{q.calib\ p_v\ (T\ jitter)} \mid) \tag{1}$$

Assuming that jitter is always positive, we can substitute the critical value $q.calib\ p_v\ (T\ jitter)$ from Figure 6 and solve for the jitter:

$$\mid\ i.error\ p_v\ (T\ jitter)\ x\ y\ t\ \mid \leq -\ln(0.75) * 0.1 \doteq 0.029 \tag{2}$$

Thus, the absolute value of the jitter can be as large as 0.029 seconds. Since all other errors are assumed to be zero, jitter is defined by the error model to be the difference $t_{ideal} - t$ where content displayed by the presentation at time t should have been displayed at time t_{ideal}. As Figure 9 illustrates, if the duration d of the ith frame in a presentation is centered on the ideal time for presentation of that frame t_i then the absolute value of the jitter is always less than $d/2$ seconds. Setting $d/2 \leq 0.029$ and solving for d gives us a maximum frame duration of 0.058 seconds and a minimum frame rate of approximately 18 frames/second.

Analysis of a QOS specification can identify a range of presentation plans that might satisfy the specification as illustrated above. A multimedia player can perform this analysis automatically

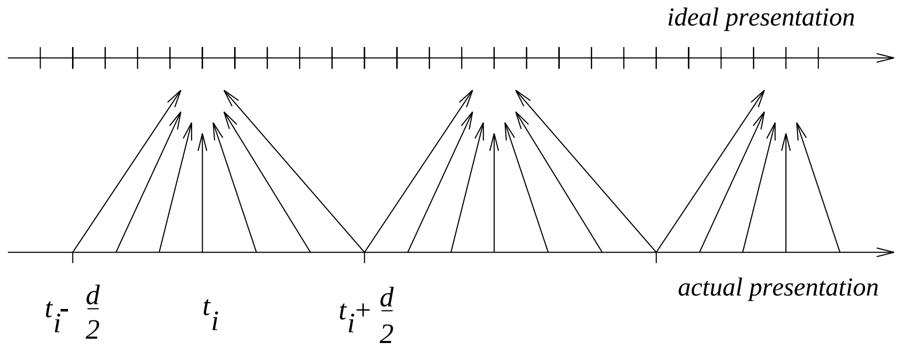

Figure 9: Example mapping from actual presentation times to ideal presentation times. When shift error in an interpretation is zero, all timing error must be attributed to jitter.

in response to playback requests. To guarantee that a particular presentation plan will satisfy a QOS specification a player must reserve resources for storage access, decompression, mixing, and presentation processes [21]. The attempt to reserve resources is called an admission test. The admission test may invoke resource reservation protocols for network and file system resources with resource-level QOS parameters derived from the process timing requirements. If the player can not find a presentation plan that both satisfies the QOS requirements and meets the admission test, then the QOS requirements must be renegotiated.

7 Related Work

It is now well understood that time-based multimedia systems require some form of resource guarantees for predictable performance. We consider related research in the categories of content specification, QOS specification, scheduling mechanisms and reservation protocols.

The Muse system [13] is one of the earliest full-featured editing tools that allows multi-track timeline synchronization of media objects. The Muse authoring tools support content and view specification, but do not explicitly constrain presentation quality. MAEstro [9] is a distributed editing and playback environment that achieves effective coarse-grained synchronization for timeline-based content specifications. The MAEstro player relies on UNIX timer interrupts, Sun remote procedure calls, and the Unix scheduler for best-effort synchronization. The CMIFed [26] environment supports content editing with specification of allowed deviations in synchronization. Our work can be applied to extend these tools with a formal model for specifying quality along with content. The QOS specifications can be used both to guarantee acceptable presentations and to optimize resources when the quality requirements admit multiple presentation plans.

Researchers have suggested a variety of parameters for multimedia QOS specifications. Continuous media stream access is generally described by throughput and delay or jitter bounds [2, 20, 25, 32]. Hutchinson, et al. [14], suggest a framework of categories for QOS specification including *reliability*, *timeliness*, *volume*, *criticality*, *quality of perception* and even *cost*. They provide only a partial list of QOS parameters to show that current QOS support in OSI and CCITT standards is severely limited. While these lists suggest many important ways to describe service categories, they go beyond presentation requirements and into specification of implementation. Our definition of QOS specification excludes volume, throughput and cost values because these values are secondary and can be derived from the combination of presentation requirements and system configuration. The Capacity-Based-Session-Reservation-Protocol (CBSRP) [32] supports reservation of processor bandwidth from the specification of a range of acceptable spatial and temporal resolutions for video playback requests. The resolution parameters are intended only for providing a few classes of service based on resource requirements and not for completely capturing presentation quality requirements.

Our error model provides a complete set of error components including *Z shift*, *Z rate*, and *Z jitter* components for image value errors as well as error components for inter-stream synchronization.

Many researchers have demonstrated that quality can be traded for lower bandwidth requirements during a presentation. A variety of scaling methods may be applied to reduce the bandwidth requirements of video streams [6, 8]. Software feedback techniques have been used to dynamically adjust stream processing workloads to available system bandwidth [5, 24, 27, 32]. These techniques can be used aggressively by a presentation planner to reserve minimal resources for a formal QOS specification.

Resource requirements may be derived from a presentation plan that satisfies a QOS specification. When the resource requirements are known, resource reservation protocols are needed to guarantee predictable access. Several groups have reported reservation protocols for network resources [1, 37, 38]. Processor capacity reservation has been implemented in the Real-Time Mach operating system [22] and file systems have been developed to support reservations for continuous media streams [3, 20, 25, 35]. These protocols can be used effectively within the architecture suggested in Section 2.

8 Conclusions

This paper has described a new framework for QOS specification in multimedia systems. The primary contributions of this framework are the clear distinction between *content*, *view* and *quality* specifications, and the formal definition of presentation quality. Because every component of our QOS specifications has an unambiguous meaning it is possible to prove the correctness of a presentation plan. These formal QOS specifications enable system designers to request and provide meaningful end-to-end guarantees for multimedia services. Section 6 gave an informal illustration of how the QOS specification can be used to derive a minimal frame rate for an acceptable presentation.

Our formal definition of presentation quality is based on a mapping from an actual presentation to an ideal specification. This mapping ensures that our error model is complete because, as all error components in the model approach zero, the presentation necessarily becomes indistinguishable from the ideal specification. Previous definitions of QOS parameters do not satisfy this completeness criteria. We have proposed an extensible set of error components that are a superset of the QOS parameters suggested by other researchers.

Another important achievement of this definition is the recognition that presentation quality should be specified in terms of a subjective interpretation of output errors and not in terms of the presentation mechanism. Our QOS specifications allow a player to choose an optimal presentation plan according to current resource costs and availability. Knowledge of the presentation mechanism can then be used to prove that an acceptable quality interpretation of the presentation exists.

We are implementing a playback system that uses these QOS specifications. Further work is needed to investigate algorithms for translating QOS specifications into feasible presentation plans. Studies of human perception are also needed to improve the error model.

8.0.1 Acknowledgements.

We would like to thank Tom Little of Boston University and John Nicol of GTE Labs for their comments on early versions of this work. Jim Hook of The Oregon Graduate Institute of Science & Technology also aided our understanding of the formalisms required.

References

[1] David P. Anderson, Ralf Guido Herrtwich and Carl Schaefer: SRP: A Resource Reservation Protocol for Guaranteed-Performance Communication in the Internet. Tech. Rep. 90-006, International Computer Science Institute, February, 1990.

[2] David P. Anderson, Yoshitomo Osawa and Ramesh Govindan: A File System for Continuous

Media. ACM Transactions on Computer Systems, Vol. 10, No. 4, November 1992, pp. 311-337.

[3] David P. Anderson: Metascheduling for Continuous Media. ACM Transactions on Computer Systems, Vol. 11, No. 3, August 1993, pp. 226-252.

[4] Apple Computer, Inc.: Inside Macintosh: QuickTime. Addison-Wesley, 1993.

[5] Andrew Campbell, Geoff Coulson, Francisco Garcia and David Hutchison: A Continuous Media Transport and Orchestration Service. Proceedings ACM SIGCOMM '92, Baltimore, MD, August 1992.

[6] Tzi-cker Chiueh and Randy H. Katz: Multi-Resolution Video Representation for Parallel Disk Arrays. ACM Multimedia 93 Proceedings, August 1993, pp. 401-410.

[7] Geoff Coulson, Gordon S. Blair and Philippe Robin: Micro-kernel Support for Continuous Media in Distributed Systems. Computer Networks and ISDN Systems, Vol. 26, 1994, pp. 1323-1341.

[8] Luca Delgrossi, C. Halstrick, D. Hehmann, R.G. Herrtwich, O. Krone, Jochen Sandvoss and Carsten Vogt: Media Scaling for Audiovisual Communication with the Heidelberg Transport System. ACM Multimedia 93 Proceedings, August 1993, pp. 99-104.

[9] G.D. Drapeau: Synchronization in the MAEstro Multimedia Authoring Environment. ACM Multimedia 93, August, 1993, Anaheim, CA., pp. 331-340.

[10] Jim Gemmell and Stavros Christodoulakis: Principles of Delay-Sensitive Multimedia Data Storage and Retrieval. ACM Transactions on Information Systems, Vol. 10, No. 1, January 1992, pp. 51-90.

[11] Rei Hamakawa and Jun Rekimoto: Object Composition and Playback Models for Handling Multimedia Data. Multimedia Systems, Vol. 2, No. 1, June 1994, pp. 26-35.

[12] B.B. Hehmann, M.G. Salmony and H.J. Stuttgen: Transport Services for Multimedia Applications on Broadmand Networks. Computer Communications, Vol. 13, No. 4, May 1990, pp. 197-203.

[13] M.E. Hodges, R.M. Sasnett, M.S. Ackerman: A Construction Set for Multimedia Applications, IEEE Software, January 1989, pp. 37-43.

[14] David Hutchison, Geoff Coulson, Andrew Campbell and Gordon S. Blair: Quality of Service Management in Distributed Systems. Tech. Rep. MPG-94-02, Lancaster University, 1994.

[15] K. Jeffay, D.L. Stone, T. Talley, F.D. Smith: Adaptive, Best-Effort Delivery of Digital Audio and Video Across Packet-Switched Networks. Proceedings of the Third International Workshop on Network and Operating System Support for Digital Audio and Video, November 1992, pp. 1-12.

[16] A. Lazar, G. Pacifici: Control of Resources in Broadband Networks with QOS Guarantees. IEEE Communications Magazine, October 1991.

[17] Didier Le Gall: MPEG: A Video Compression Standard for Multimedia Applications. CACM, Vol. 34, No. 4, April 1991.

[18] T.D.C. Little and A. Ghafoor: Network Considerations for Distributed Multimedia Object Composition and Communication. IEEE Network Magazine, November 1990, pp. 32-49.

[19] T.D.C. Little, A. Ghafoor: Interval-Based Conceptual Models for Time-Dependent Multimedia Data. IEEE Transactions on Knowledge and Data Engineering, Vol. 5, No. 4, August 1993, pp. 551-563.

[20] P. Lougher and D. Shepherd: The design of a storage server for continuous media. The Computer Journal, Vol. 36, No. 1, February 1993, pp. 32-42.

[21] David Maier, Jonathan Walpole and Richard Staehli: Storage System Architectures for Continuous Media Data. FODO '93 Proceedings, Lecture Notes in Computer Science, Vol. 730, 1993, Springer-Verlag, pp.1-18.

[22] C.W. Mercer, S. Savage and H. Tokuda: Processor Capacity Reserves: Operating System Support for Multimedia applications. Proceedings of the International Conference on Multimedia Computing and Systems, May 1994, pp. 90-99.

[23] Jason Nieh, James G. Hanko, J. Duane Northcutt and Gerard A. Wall: SVR4UNIX Scheduler Unacceptable for Multimedia Applications. Proceedings of the 4th International Workshop on Network and Operating System Support for Digital Audio and Video, November 1993, pp. 35-47.

[24] Calton Pu and Robert M. Fuhrer: Feedback-Based Scheduling: a Toolbox Approach. Proceedings of Fourth Workshop on Workstation Operating Systems, October 1993.

[25] K.K. Ramakrishnan, Lev Vaitzblit, Cary Gray, Uresh Vahalia, Dennis Ting, Percy Tzelnic, Steve Glaser and Wayne Duso: Operating System Support for a Video-On-Demand File Service. Proceedings of the 4th International Workshop on Network and Operating System Support for Digital Audio and Video, November 1993, pp. 225-236.

[26] G. van Rossum, J. Jansen, K.S. Mullender, and D.C.A. Butlerman: CMIFed: A Presentation Environment for Portable Hypermedia Documents. ACM Multimedia 93, pp. 183-188.

[27] Lawrence A. Rowe, Ketan D. Patel, Brian C. Smith, Kim Liu: MPEG Video in Software: Representation, Transmission, and Playback. High Speed Networking and Multimedia Computing, IS&T/SPIE, February 1994.

[28] J.M. Spivey: The Z Notation: A Reference Manual. Second edition, Prentice-Hall International, 1992.

[29] Richard Staehli and Jonathan Walpole: Constrained-Latency Storage Access. Computer, Vol. 26, No. 3, March 1993, pp. 44-53.

[30] Ralf Steinmetz and Clemens Engler: Human Perception of Media Synchronization. Tech. Rep. 43.9310, IBM European Networking Center, 1993.

[31] Andrew S. Tanenbaum: Computer Networks – 2nd Edition. Prentice-Hall, 1988, pp. 278.

[32] Hideyuki Tokuda, Yoshito Tobe, Stephen T.-C. Chou and José M.F. Moura: Continuous Media Communication with Dynamic QOS Control Using ARTS with an FDDI Network. SIGCOMM '92, 1992.

[33] Hideyuki Tokuda and Takuro Kitayama: Dynamic QOS Control based on Real-Time Threads. Proceedings of the 4th International Workshop on Network and Operating System Support for Digital Audio and Video, November 1993, pp. 113-122.

[34] Hideyuki Tokuda: Operating System Support for Continuous Media Applications. In Multimedia Systems, Ch. 8. Addison-Wesley, 1994, pp. 201-220.

[35] H. M. Vin and P. V. Rangan: Designing a Multi-User HDTV Storage Server. IEEE Journal on Selected Areas in Communication, Vol. 11, No. 1, January 1993.

[36] Gregory K. Wallace: The JPEG Still Picture Compression Standard. CACM, Vol. 34, No. 4, April 1991, pp. 30-44.

[37] Marek Wernik, Osama Aboul-Magd and Henry Gilbert: Traffic Management for B-ISDN Services. IEEE Network, September 1992, pp. 10-19.

[38] H. Zhang and D. Ferrari: Improving Utilization for Deterministic Service in Multimedia Communication. Proceedings of the International Conference on Multimedia Computing and Systems, May 1994, pp. 295-305.