Applying Statistical Process Control to the Adaptive Rate Control Problem

Nelson R. Manohar, Marc Willebeek-LeMair
IBM T. J. Watson Research Center
Yorktown Heights, NY 10598
{nelsonr,mwlm}@watson.ibm.com

Atul Prakash
Department of Electrical Engineering and Computer Science
University of Michigan, Ann Arbor, MI 48109
aprakash@eecs.umich.edu

Abstract

We present a framework for the streaming of heterogeneous media. We introduce the use of online statistical process control (SPC) for dynamic rate control of the streams. In SPC, the goal is to establish and preserve a state of statistical quality control (i.e., small variability around a target mean) over a process. We consider the streaming of multimedia content over the internet as our process. First, at each client, we measure process performance and apply statistical quality controls with respect to application-level requirements. Then, we guide the adaptive rate control (ARC) problem based on the statistical significance of performance trends and departures. We show this scheme to remove media-awareness from the SPC feedback mechanism and to facilitate handling of heterogeneous media types. Last, by definition, SPC is designed to monitor long-term process performance; we show that online SPC could be used as a mechanism to adapt to various degrees of long-term network variability (i.e., statistically significant process shifts as opposed to short-term process fluctuations). We develop several examples of its use and analyze its statistical behavior and guarantees.

1 Introduction

The size, heterogeneity, and complexity of the internet makes performance analysis hard. The streaming of media across the internet is subject to so many variables that significant research activity has focused on "probe-and-adapt" protocols for end-to-end delivery of multimedia streams [11, 5, 12]. To adapt, we must first probe (i.e., collect and analyze performance measurements). To probe, we must first know what to look for. To probe-and-adapt without an underlying model can lead to adjustments found later to be insufficient or worse counterproductive. Thus, we are interested in robust online models for the support of the measurements and adaptive rate control problems.

End-to-end streaming of multimedia is subject to network variability.[1] Recent observations about spatial and temporal stability [2] of end-to-end network variability suggest the presence of long-term trends. At one extreme we have a large number of protocols that address short-term fluctuations (typically measured in milliseconds) while at this other extreme we are interested in protocols to address long-term variability. However, the window size in relation to the long-term outlook influences how fast a probe-and-adapt mechanism can react to changes on variability. Too small a window, results in too frequent measurement overheads and possibly, unnecessary adjustments. Too large a window, slows down the responsiveness of the mechanism to changes in network variability.

[1] — Throughout this paper we assume that network variability is a random variable. However, we make no assumption about its distribution, as it does not affect our analysis. We discuss this in Section 2.

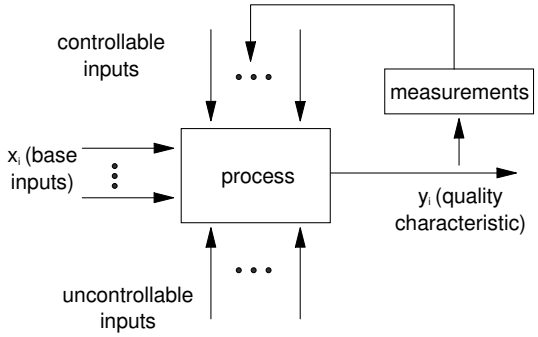

Figure 1. Block diagram of the generic production process model. There are three kinds of inputs: base inputs, controllable inputs, and uncontrollable inputs. A measurements module evaluates the process performance based on a quality characteristic and feedback its evaluation to the process as an input itself.

We want to develop mechanisms to detect and adapt to long-term changes in network variability (i.e., large window size). However, we want these mechanisms to have fast response time (i.e., small window size). In this paper, we present a best effort mechanism to measure and adapt to long-term variability in end-to-end delivery of media events. Our approach provides online characterization of long-term variability. This is achieved through the application of statistical process control (SPC) over a long-term window into process performance. The approach couples statistical measurements to adaptive rate control by relating the statistical significance of these trends and departures to the adaptation effort.

Fig. 1 shows the block diagram of a generic process model. It is composed of three kinds of inputs: base inputs such as parts, controllable inputs such as fine tuning parameters, uncontrollable inputs such as noise. A measurements module evaluates the process performance based on a quality characteristic and feeds back its evaluation to the process as an input itself. We set up the long-term adaptive rate control over a process as follows. Let $y_i = g(x_i)$ be a process defined over a time series input x_i. Let $m(y_i)$ be a performance measurement function associated with process y_i. Let t_i be a time series representing performance measurements $t_i = m(y_i)$. Last, let t_i^* be a t_i that is subject to uncontrollable inputs η_i (such as noise or random variability). Equivalently,

$$y_i = g(x_i) \tag{1}$$

$$t_i = m(y_i) \tag{2}$$

$$t_i^* = t_i + \eta_i \tag{3}$$

Our goal is twofold. First, to use t_i^* to monitor and characterize long-term trends and departures introduced by the uncontrollable inputs η_i (a SPC problem). Second, to use this knowledge to adapt the performance of process y_i to such trends (an ARC problem). To apply this model to the process of streaming multimedia content over a network let $y_i = g(x_i)$ be the process of streaming media frames x_i from a server to a client; $m(y_i)$ be a presentation measurements function (at the client) for frame x_i; and η_i be end-to-end variability incurred over the delivery and presentation of frame x_i. Our goal is now to adjust the performance of y_i (streaming of media frames) to the presence of long-term trends in network and client variability η_i.

The adaptive rate control problem is typically approached so that measurements are often defined with respect to the performance of network delivery. For example, the performance of the streaming could be defined in terms of available buffering of packets at time t subject to the goal that $buffer(t)$ never falls below a threshold. The deficiency of this model is that it assumes that the client is capable of presenting the data on time as long as the network delivers it on time. Often that is not the case. In the adaptive rate control problem we present in this paper, the performance of the network streaming is modeled in terms of application-oriented quality characteristics. The rationale behind this follows from the following process control metaphor to the streaming of multimedia content. We are interested in the measuring the capability of a *process* (end-to-end streaming) to deliver *goods or items* (media frames) *as promised* (statistical process performance guarantees)

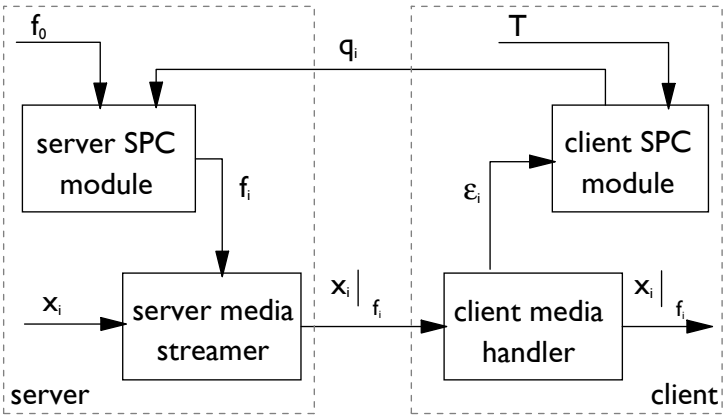

Figure 2. Integrating SPC into the streaming model between server and client. Both client and server are enhanced with SPC modules. The client SPC module monitors a quality characteristic ϵ_i with respect to client tolerance T to then produce a quality assessment q_i. The server SPC module uses the quality assessment q_i to modify a degree of freedom f_i. The degree of freedom f_i is used to influence the streaming of media frames x_i (i.e., represented as $x_i|_{f_i}$).

to *customers* (applications). An application is not interested in fractional goods, i.e., is not aware that these goods are made up of *parts* (network packets). The application's primary concern is whether or not a good was delivered as promised.

Intuitively, the process of streaming media over a network is visualized as follows. Each media server is modeled as a *supplier* of *consumable samples* (media packets). In turn, each client is modeled as a *consumer* of samples. A *distribution network* (wide area network) connects suppliers and consumers. A supplier delivers samples to a consumer over a virtual *production line* (streaming connection) of the distribution network. The production-line model asserts the fact that a *continuous supply* (delivery/streaming) of samples is given to a consumer. Independent production lines are associated each with different *types of consumables* (media types). Consumable samples are assembled at the client site into *goods* (media frames). Since production lines are often unreliable and/or subject to variable conditions, a consumer associates a *measurement process* (timing/performance measurements) with each production line so as to monitor their performance. A consumer verifies performance measurements against its own *specifications* (application-oriented tolerances). Typically, a consumer is interested in two kinds of measurements: (1) measurements that assess the quality of individual goods delivered to the consumer (e.g., these are indicators such as the number of "fit-for-use" frames per second delivered to the application) and (2) measurements that assess the quality of the supplier itself and its production process (e.g., these are indicators that characterize how reliable or good is the process of streaming between the server and client). This asserts a consumer dilemma: even the best suppliers may occasionally slip (e.g., due to uncontrollable inputs such as short-term fluctuations) and not meet customer targets while poor suppliers may occasionally meet customer targets (e.g., only under certain conditions). That is, we must look at both the process mean and its variability in order to distinguish between good and bad suppliers.

Our goal is to characterize the delivery of goods (media frames) from supplier (server) to customer (client) and manage a delivery process (streaming of multimedia between two sites) into a state of statistical quality control (small variability over a process mean that is centered on its target). To this end, we show a network feedback mechanism for the support of online statistical process control over adaptive streaming of media between a server and a client.

The rest of the paper is organized as follows. First, we describe the framework for the application of statistical process controls in the multimedia networking domain and analyze its properties. Then, we describe

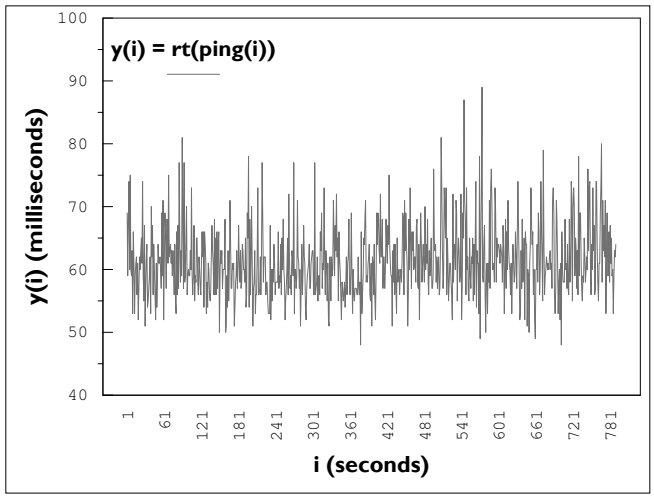

Figure 3. Adapting to network variability under a short-term view can suffer from near-sightedness. Shown is the process $y_i = roundtrip(x_i)$, where $x_i = ping()$ between two sites (umich & unc), plotted for a period of 12 minutes with samples taken every second. Roundtrip time y_i is subject to network variability and had $\mu_y = 61.3$ and $\sigma_y = 5.9$.

our experience with SPC on multimedia networking with several detailed examples and then analyze in detail its benefits. Last, we present our concluding remarks.

2 Statistical Process Controls

Let's provide intuition into the use of long-term mechanisms.

Example: We desire to design a process to reliably detect and measure long-term network variability between two sites connected through a wide-area network. To apply the process model, we need to (1) define the process $y_i = g(x_i)$, (2) define the controllable inputs x_i, (3) define the uncontrollable inputs η_i, and (4) define the measurement $t^*(y_i)$. To this end, let $y_i = g(x_i)$ be the process of sampling network delay through the use of x_i, where x_i is a ping() probe between sites 1 and 2. Each probe returns a measurement $t^*(y_i)$ of the roundtrip delay between sites 1 and 2. Each probe measurement is subject to random network variability (η_i).

Although network variability can not be controlled, SPC can still be applied to monitor network variability and to reliably (i.e., with known statistical confidence) detect when the network variability shifts into a different process mean. This can be viewed as a sort of temporal stability monitor. Fig. 3 illustrates a time series plot for the process $y_i = g(x_i)|_{i=i..760}$, where x_i = the i^{th} ping() command between two particular sites. Under a short-term outlook, the data appears to have a stable mean subject to random jitter. However, under our notion of long-term outlooks, three distinct process shifts as well as a steadily increasing process mean (center line) emerge from the same time series. Fig. 4 shows the results of our long-term analysis based on a two minute outlook created by computing the sample mean and standard deviation with a moving window of size $m = 120$ seconds.

Naturally, the next question is whether we can design mechanisms to take advantage of this knowledge. We present an end-to-end measurement feedback model to compensate for long-term process performance under statistical guarantees. For each media type, a server manages an adaptive rate control (ARC) problem with respect to a degree of freedom resource associated with the media type. Similarly, for each media type, a client collects measurements and applies to these statistical quality controls (SQC) to detect the presence of statistically significant trends in process performance. A feedback loop couples client to server and it is used

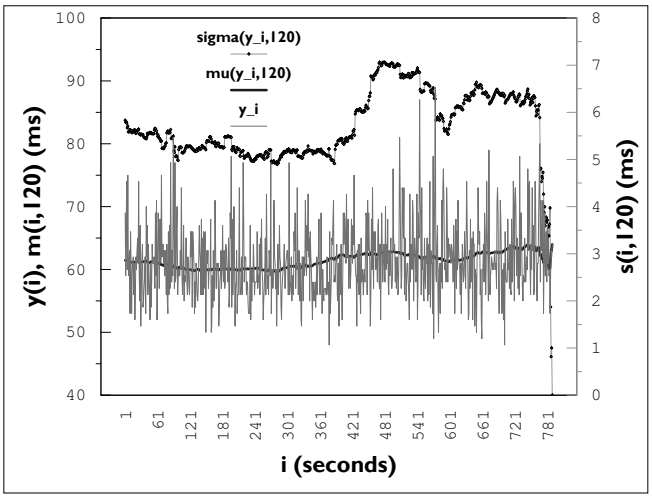

Figure 4. Stepping out of the woods and seeing the forest. With a long-term moving window of 120 seconds into the process performance, we examine $\mu(t_{i=k..k+120})$ and $\sigma(t_{i=k..k+120})$. Long-term trends emerge. Specifically, the network variability exhibited three different plataous while its process mean exhibited a steady increase.

to relay process feedback into the ARC problem for this media type at the server. The coupling of ARC with SQC attempts a best-effort compensation over the statistical performance of the streaming process between server and client.

An instance of the framework in its simplest form is shown in Fig. 5. It shows a media server (S) and a client (C). The media server streams a media type (M) to a client. Periodically, during the presentation of a media stream, the client collects scheduling measurements (ϵ_i). Scheduling measurements can be derived from either temporal (progress against time) or relative (progress against another stream) time systems. The client applies statistical quality controls (SQC [1]) over an smoothed time series ϵ_i^* against a stream tolerance T. In a nutshell, SQC provides robust statistical analysis of performance departures from a target or process mean (for example, $(\epsilon_i - 0)$). These departures are then compared against either target tolerances (say T) or smoothed past process variability (say σ_i^*). This SQC analysis at the client produces a feedback report $\Delta(\epsilon_i^*, \sigma_i^*)$ that ranks the severity of statistically-significant (i.e., sufficiently large) performance departures.

The $\Delta(\epsilon_i^*, \sigma_i^*)$ contains enough information to rank performance departures into two types: yellow and red flags. A yellow flag represents departures deemed significant by the client SQC and compensable by the server ARC. A red flag represents departures typically associated with a (streaming) process lacking a state of statistical process control (the absence of small and consistent variability ($\sigma_i^* \to 0$) over the target mean of $\epsilon_i^* \to 0$). A yellow flag triggers a process control action at the media server which can result in an adjustment over the media's degree of freedom. The goal of this adjustment is to induce change on the process performance of the streaming process. A red flag requests that the media server resets its streaming to the client. The goal of the red flag is to indicate that process performance can not be met under current parameters and that the streaming must be restarted under less stringent parameters or robust degree of freedom. The feedback reports $\Delta(\epsilon_i^*, \sigma_i^*)$ are sent to the server through a reliable TCP/IP connection to the media server. The media server assumes that the absence of a report implies a state of process control at the client.

To manage heterogeneous media, the framework decouples server and client; the server streams media whereas the client provides statistical measurements of the end-to-end performance of the streaming process with respect to application-level requirements. The framework intends to deliver a best effort approach. It relies on a global feedback loop that is coupled with statistical process control so as to induce long-term properties over the accuracy of the feedback and forecast. Note that the time span of our long-term process control iteration (seconds or minutes) is several orders of magnitude greater than the delays associated with a global feedback loop (milliseconds).

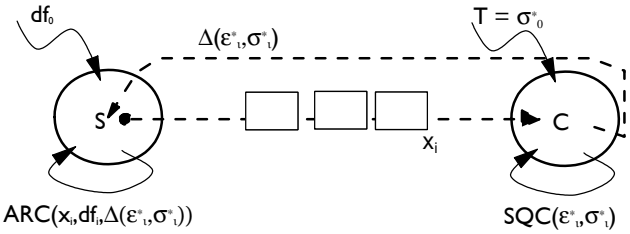

Figure 5. Extending SPC to the ARC problem. The ARC is formulated over a media type and managed by the server. The client collects measurements and applies statistical quality controls to detect statistically significant trends. A feedback loop couples server and client is used to relay process feedback into the ARC problem.

To illustrate the use of this framework, consider a server that streams audio (A) and video (V) used for lip-synchronous playback at a client. Using temporal streaming, the performance of the streaming to these streams could be measured as asynchrony (or drift) against the system clock (LTS) of the client. Each performance measurement at the client results in time-series $\epsilon_i(A, LTS)$ and $\epsilon(V, LTS)$. Each time series is independently analyzed by a statistical client with respect to the tolerance of the client to process departures. In this example, we desire to use SPC to monitor, establish, and preserve the asynchrony trends and departures to within the tolerance of lip-synchronous playback. To do so, the client analyzes each $\epsilon_i()$ time series to assess whether each streaming process lacks statistical process control. In such cases, the client sends a feedback report $\Delta(\epsilon^*, \sigma_i^*)$ to the server. The server uses such feedback to adjust the performance of its streaming by triggering a compensation over the degree of freedom associated with the stream at fault. The compensation is computed based on the statistical significance of the performance departure (i.e., smoothing is subject to the goal of statistical process performance). Under this scenario we could use a variable sampling rate as the degree of freedom for the audio stream and similarly, use a variable frame rate as the degree of freedom for the video stream. The block diagram for the statistical analysis performed by the client is shown in Fig. 6.

To retroffit statistical process controls into media servers, we model the media server as being composed of: (1) the original media streamer (which handles the fetching, scheduling, and streaming of media frames), and (2) our ARC module (which adapts the statistical performance of the streaming). The ARC module issues adaptive recommendations to the media streamer. Similarly, we model the client as being composed of: (1) the original media handler, and (2) our SQC measurements module. The SQC module issues statistical process assessment to the ARC module. This architecture is shown in Fig. 2 and in Fig. 5. Next, we formally define these server and client SPC modules.

The ARC problem is defined as follows. First, to each media stream m, we associate a degree of freedom resource df. A few examples of adaptable degree of freedom resources are frame rate for video, sample rate for continuous audio, elastic silence for discrete audio, elastic time for asynchronous streams, and quantization error for compressed video. To each media stream, we associate an ARC problem with parameters: (1) df: the degree of freedom; (2) f_i: the adaptation effort at the i^{th} iteration; and (3) $\Delta(\epsilon_i^*, \sigma_i^*)$: a ranking of the process feedback at the i^{th} iteration.

$$f_i = ARC(f_{i-1}, \Delta(\epsilon_i^*, \sigma_i^*)) \tag{4}$$

The process feedback rating function $\Delta(\epsilon_i^*, \sigma_i^*)$ represents the statistical significance of process feedback as a tuple $< r, p >$, where r is the strength of the performance departure when measured in standard deviations σ_i^* and p is the $1 - \alpha$-confidence on this assessment. For example, assuming the indicator ϵ_i^* is a normally distributed random variable, we define $\Delta(\epsilon_i^*, \sigma_i^*)$ to be:

$$\Delta(\epsilon_i^*, \sigma_i^*) =< r_i, p_i > \tag{5}$$

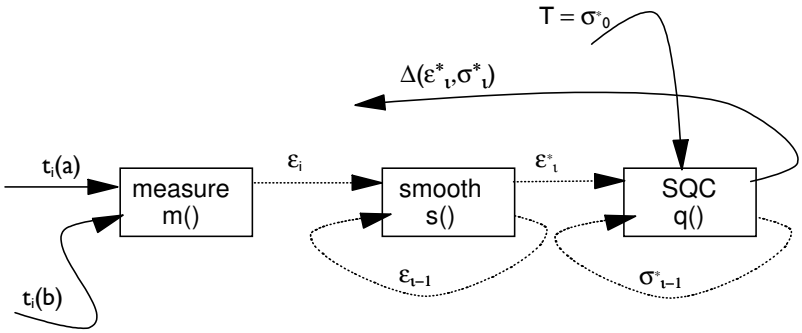

Figure 6. The client SPC module provides long-term performance measurements and assessment over a quality characteristic, being in this case, the asynchrony ϵ_i. The SPC module consists of three components: (1) measure() which computes ϵ_i for either temporal or relative scheduling, (2) smooth() which smooths ϵ_i into ϵ_i^*, and (3) SQC() which applies statistical quality controls to the smoothed time series ϵ_i^*.

where r_i is a positive integer and p_i is given by $p_i = (1 - P(|\epsilon_i^*| > r_i * \sigma_i))$. Under a normality assumption on ϵ_i^*, p_i is easy to compute.

A yellow flag is raised by each feedback report. The red flag, on the other hand, is raised when the time series $(r_i, r_{i-1}, \cdots)$ exhibits out-of-control conditions. For example, note that the presence of a trend or a run on this time series warns that a statistically significant shift in process performance is occurring. In SPC, the detection of process shifts is associated with conditional probabilities about the likelyhood of having patterns of statistical exceptions. To illustrate this, assume that the server has seen so far the time series $r_i = (3, 3, 3, \cdots)$. We associate such trend or process shift with the likelyhood of having three statistical exceptions, of strength $3\sigma_i$, in a row.

When a feedback report is received, the degree of freedom f_i is updated to compensate for process departures. The compensation effort is given by $f_i - f_{i-1}$. Note that the rankings r_i correspond to cumulative probabilities p_i so the compensation effort associated with each ranking r_i should factor these unequal weights. One approach is to define the compensation effort $(f_i - f_{i-1})$ so that is proportional to the strength of the performance departure as:

$$\Delta_f(i) = f_i - f_{i-1} = p_i(\epsilon_i^* \pm \sigma_i^*) \tag{6}$$

This has the benefit of tailoring the aggressiveness of the compensation effort to the confidence over the performance departure. Why? Our knowledge about the relationship of the true mean of the asynchrony μ_ϵ with respect to the indicator ϵ_i^* is given by the confidence interval $(\epsilon_i^* - p_i * \sigma_i^* \leq \mu_\epsilon \leq \epsilon_i^* + p_i * \sigma_i^*)$; where p_i equals $Z_{\alpha/2}$ if σ_i^* is stable or $t_{\alpha/2,m-1}$ otherwise. With confidence $100(1-\alpha)$ we know that the true mean of the asynchrony lies within that interval (for example, Fig. 9 gives a graphical illustration of the accuracy of the 80% confidence region and the ϵ_i^* indicator with respect to the original ϵ_i time series). But because we don't know where exactly the mean lies, we tailor the strenght of the compensation effort to the confidence we have on the assessment. To illustrate this, suppose that our current ARC parameters are $< r_i = 2, p_i = 0.80, \epsilon_i^* = 100ms, \sigma_i^* = 5ms >$. These parameters can be read as follows: "a streaming process y_i has been delivering media frames with a skew of $(100 \pm 5ms)$ which resulted in a statistical exception of strenght $2\sigma_i^*$ (at a confidence level of of 80%) with respect to past process performance." This results in approximately an adjustment of $\Delta_{f_i} = 80\frac{ms}{s}$ needed to compensate a (possible) trend of asynchrony of $100ms$.

This is not the only possible compensation strategy to reducing the asynchrony ϵ_i^*. It illustrates however, the basic idea that the compensation can be made on our confidence on the true strenght of a performance departure. A refinement is to realize that our long-trend outlook effectively smooths asynchrony into piece-wise linear segments that are either horizontal or with a slope (for example, see Fig. 10). A horizontal segment represents asynchrony that is stable (that is, the asynchrony ϵ_i^* for the streaming between server and client

has been stable at this level). For such segments, we want to amortize the compensation over the asynchrony ϵ_i^* so as to gradually return the mean of the asynchrony to its target (that is, $\epsilon_i^* \to 0$). On the other hand, a segment with a slope represents a mean that is building up (or decreasing, depending on the slope). For such segments we want to counteract by delivering continuous compensation until the mean gets stabilized (becomes a horizontal segment). All the information needed to detect a process shift (horizontal line) or a build-up (a slope) on the asynchrony is available in our ARC tuple(s). We would like to compare several long-term compensation effort strategies and report their performance in this paper.

To support multiple media types, we defined for each media type a compensation constant c between 0 and 1. The compensation constant is a weight factor that specifies how to distribute compensation effort for a given media type (i.e., $c * \Delta_{f_i}$). Its default value is one. Small values of c result in lesser weight to process control compensations for this particular stream. Larger values of c result in more weight to the process control compensations. The updated degree of freedom resource $f_i = f_{i-1} + \Delta_{f_i}$ is used by the media handler to adapt the performance of its streaming. Let x_j be the first media frame after compensation f_i is computed. The streaming of this and subsequent media frames $[x_j, x_{j+1}, \cdots]$ is adapted to this new compensation effort f_i. The compensated streaming of x_j is denoted as $x_j|_{f_i}$. To illustrate this, consider a non-compressed video stream using the frame rate as its degree of freedom df with current value $f_{i-1} = 30fps$. Suppose that our current ARC tuple is $< r_i = 2, p_i = 0.80, \epsilon_i^* = 100ms, \sigma_i^* = 5ms >$. We compute the compensation effort to be in the range of $\Delta_{f_i} = [90.25, 99.75]\frac{ms}{s}$. Since each video frame carries a $33ms$ degree of freedom, we need approximately $\frac{[90.25,99.75]}{33}$ frames to meet our compensation effort (the constant c is used here to adjust this mapping). Consequently, the updated frame rate for the video stream becomes $f_i = [27, 28]fps$.

Given two streams x and y, their relative process performance indicator is:

$$\epsilon_i = t(x_i) - t(y_i) \tag{7}$$

where y can be the system clock LTS. To achieve resiliency over random fluctuations, we smooth the process performance indicator, by factoring past performance indicators $\epsilon_{i-1,i-2,\ldots}$. We have several choices on how to smooth. An exponential weighted forecast with geometrically decaying weights (for example, $\epsilon_i^* = \lambda\epsilon_i + (1-\lambda)\epsilon_{i-1}$) is very useful for its fast response to small process shifts. The EWMA SQC-chart [4, 9] is used to control this smoothed time series. For simplicity and because of its statistical properties, we chose to smooth using a moving average (MA) of window size m (that is, $\epsilon_i^* = \frac{\sum_{k=i-m}^{i}}{m}$). By the *Central Limit Theorem*, we know that, regardless of the distribution of the population, the distribution of $\bar{x} = \sum_{i=1}^{n} x_i$ is approximately normal with mean $n\mu$ and variance $n\sigma^2$. That is, regardless of the distribution of ϵ_i, the distribution of its moving average is normally distributed with mean ϵ_i^* and standard deviation $\sigma_i^* = \frac{\sigma}{m}$. That is, our assumption of normality on ϵ_i is met and since ϵ_i is a random variable derived from network variability, we do not need to know the distribution of the network variability.

The statistical quality control [10] over the performance of the streaming is formulated at the client as:

$$\Delta(\epsilon_i^*, \sigma_i^*) = SQC(\epsilon_i^* \to 0, (T - \sigma_i^*) \to 0) \tag{8}$$

which monitors the behavior over time of the time series ϵ_i^* over time with respect to its stability (small variance in σ_i^* and small mean asynchrony ϵ_i^* is ideal) and its ability to meet the specified tolerances (variability of at most of T and mean asynchrony centered around zero). A detailed discussion of SQC is found in [10].

2.1 Sensitivity Enhancements

A characterization is not useful in an online system if it takes too long. Some questions arise here. SQC can be parametrized to different levels of sensitivity to process shifts and performance departures and different average reaction time to these. Both properties have been studied extensively elsewhere [1, 3, 10].

What is the periodicity of the measurement system? In our experiments so far, we have used a periodicity of 1 and 2 seconds. If the periodicity is too small, our feedback mechanism incurs in too much measurements overhead. If the periodicity is too large, the feedback mechanism takes too long a time to react. How fast does SQC reacts to a process shift (changes in the mean)? This is characterized by the Average Run Lenght (ARL) of a control chart [10]. The ARL specifies the average number of samples between sampling and detection of a process shift in the mean. The ARL for a control chart is determined by: (1) the sensitizing rules used to test for outliers, (2) the process indicators used to control the process, and (3) the forecast strength of the

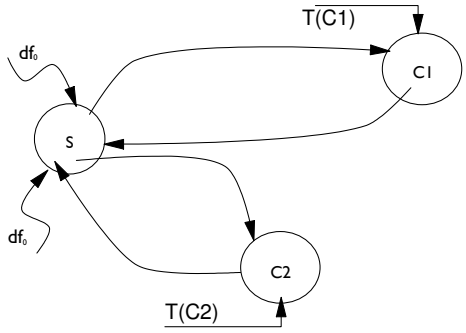

Figure 7. Streaming a presentation to multiple clients. For the same presentation, each client defines different tolerances suited to its platform capabilities and performance. The tolerances facilitate accounting for client heterogeneity.

smoothed process indicators. There are two basic approaches to reducing the ARL of a control chart: (1) by applying more aggressive sensitizing rules and (2) by using more robust process indicators. The Western Electric sensitizing rules for enhancing the detection of outliers [3] are: (1) one point outside the 3σ control limits; (2) two out of three consecutive points beyond the 2σ warning limits; (3) four out of five consecutive points beyond the 1σ limits. In this paper, we relied on the moving average control chart, a robust control chart for detecting small process shifts in the mean. To further enhance the robustness of the forecast, we could rely on an exponential forecast with geometrically decaying weights version of the moving average control chart (EWMA) [4, 9].

2.2 Temporal Streaming

This scenario is typical of the streaming of continuous media such as the streaming of two video streams, say V_1 and V_2. Each stream is associated with a different degree of freedom. For example, let $df(V_1)$ be the frame rate and let $df(V_2)$ be a compounded resource such as (frame rate coupled with quantization and buffer smoothing). Independent tolerances are specified over the process performance of each of the streaming of each of these video streams (for example, $T(V_1) = 100ms$ and $T(V_2) = 250ms$). In temporal streaming, we measure the performance of the streaming as skew $\epsilon_i(V, LTS)$ of the video stream against the system clock. For each stream V, we apply statistical quality controls:

$$SQC(\epsilon_i^*(V, LTS) \rightarrow 0, (T(V) - \sigma_i^*(V)) \rightarrow 0) \tag{9}$$

to analyze $\epsilon_i^*(V, LTS)$ with respect to the client's tolerance $T(V)$. The streaming of more than two streams against the system clock is an obvious extension to the temporal streaming of two streams. Temporal streaming to multiple clients is feasible whenever variability on the network time server between client and server can be assumed negligible with respect to the application tolerances $T(V)$.

2.3 Relative Streaming

We also need adaptive rate controls capable of addressing the impact of heterogeneity of media types over network streaming. Research on adaptive protocols have focused mostly on temporal streaming of continuous media. On the other hand, asynchronous media requires the use of relative streaming (that is, scheduling with respect to the progress of other streams). Adapting the rate at which to stream asynchronous media is now complicated by variability during both delivery and presentation of asynchronous media events. One such scenario is the streaming of audio-annotated window events, say A and W. Stream A would be associated with f_A = (adaptive sampling rate) while W would be associated with f_W = (elastic inter-event delays) as described in [8]. Since relative streaming decouples the logical time systems of both streams, a binary

relationship $R(A,W)$ is defined between streams A and W to preserve their correspondence during their relative streaming. Process performance is measured as the asynchrony between the two streams in the time series $\epsilon_i(A,W) = t_i(A) - t_i(W)$. To each of the two possible asynchrony conditions (A ahead of W; W ahead of A), we assign a tolerance T_A and T_W, respectively. For both streams, we apply statistical quality controls in the form of:

$$SQC(\epsilon_i^*(A,W) \rightarrow 0, (T(A,W) - \sigma_i^*(A,W)) \rightarrow 0) \tag{10}$$

This SQC analyzes the smoothed asynchrony $\epsilon_i^*(A,W)$ with respect to the tolerance $T(A,W)$ for their relationship.

Relative streaming of more than two binary relationships is not an obvious extension to the relative streaming of two streams. If guaranteed simultaneous presentation to multiple clients is not important, relative streaming to more than one clients is an obvious extension to the relative streaming to a single client and it is illustrated in Fig. 7. In this short paper we do not address guaranteed simultaneous presentation to multiple clients.

2.4 Framework

The notion of long-term measurement presented here does not depend on a particular media integration framework. However, below we present a framework we have designed to specifically support temporal and relative streaming of heterogeneous media in a way that is compatible with our process control model for adaptive rate control.

A synchronization event $s_i(R(\cdots))$ is used to enforce a time equals synchronization relationship across streams. Whenever an exact temporal ordering across streams in a relationship $R(\cdots)$ is needed, the synchronization event $s_i(R)$ is used. A scheduling interval is delimited at each end by synchronization events. A synchronization relationship $R(a_i \leftarrow a_j) : policy$ characterizes the nature of the synchronization relationship between two streams a_i and a_j in terms of (1) a master-slave dependency constraint (i.e., the playback of a_j synchronizes to a_i); and (2) a treatment policy for the handling of the dependency constraint (i.e., the schedule of a_j adapts to a_i). For example, the synchronization relationship $R(audio \leftarrow gestures) : policy$, where *policy* prescribes "*gestures are to adapt to the unconstrained playback of audio*" represents an end-user characterization of the playback of audio-annotated gesturing.

A synchronization event $s_i(R(a_1 \cdots a_m))$ is used to request point synchronization between scheduling intervals $T_i(a_1) \cdots T_i(a_m)$ of different streams involved in a synchronization relationship $R(a_1 \cdots a_m)$. A synchronization event $s_i(R(lhs \leftarrow rhs))$ is implemented as one or more pair-wise (binary) dependency constraints between each slave stream in the rhs of $R(\cdots)$ (those on the right hand side of the arrow) and their master stream lhs (the left hand side). For example, the synchronization relationship $R(a_j \leftarrow a_k a_m)$ (read as "*streams (a_k & a_m) synchronize to stream a_j*") leads to the following two pair-wise dependency constraints:

$$\left\{ \begin{array}{l} d_{jk}(a_j \rightarrow a_k) \\ d_{jm}(a_j \rightarrow a_m) \end{array} \right\} \tag{11}$$

A dependency constraint $d_{jm}(a_j \rightarrow a_m)$ is read as "*stream a_j releases stream a_m*" and specifies that when a_m is ahead of stream a_j, stream a_m is to "*meet*" with stream a_j for each corresponding synchronization event across the two streams. As a result, whenever stream a_m fails to "*meet*" a_j at a synchronization event, then a synchronization treatment *treat* is applied over a_m (a_m waits, when ahead of a_j). Dependency constraints are asymmetrical (e.g., a_j does not wait for a_m when a_j is ahead). The asymmetry of a dependency constraint d_{jm} can be defined in terms of the playout times of synchronization events $s_i R(a_j \rightarrow a_m)$ as follows:

$$d_{jm}(a_j \leftarrow a_m) = \left\{ \begin{array}{l} t_{s_i(a_j)} > t_{s_i(a_m)} : treat(a_j); \\ t_{s_i(a_j)} \approx t_{s_i(a_m)} : nil; \\ t_{s_i(a_j)} < t_{s_i(a_m)} : nil; \end{array} \right\} \tag{12}$$

where *treat* specifies the synchronization treatment to be applied over a_m for correcting the playback asynchrony between a_j and a_m (that is, corrected only when a_m is ahead of a_j). We refer to this scenario as a_m failing to "*meet*" its dependency constraint with a_j. The notion of a pair-wise "*meet*" of streams is central to our approach and it is discussed below.

Different synchronization relationships specify different ways of satisfying the "*meet*" constraint through the *policy* specification. Let i be the i^{th} synchronization event of stream a_j and i^* be its matching (i.e., i^{th}) synchronization event on another stream a_m. There are three basic meeting policies that specify how i and i^* *ought to meet* during the playback of these two streams. These are:

- absolute: $i = i^*$
  that is, the synchronization points i and i^* must be replayed at the same time.
- lazy: $i - \epsilon < i^* < i + \epsilon$
  that is, the synchronization points i and i^* can be replayed relatively close to each other.
- laissez-faire: $k = i^*$
  that is, the synchronization points i and i^* can be replayed in any sequence.

A *policy* specification for a relationship specifies one of the aboves options. A treatment *treat* represents the run-time enforcement of a pre-selected policy. Not all "*meet*" policies are the same. Meet policies can be ordered in terms of the strength of their tolerance of asynchrony departure between synchronization points i and i^*. Equivalently, absolute > lazy > laissez-faire

Between any two streams a_j and a_m, there are three different types of dependency constraints: 2-way, 1-way, and 0-way. In a 2-way dependency constraint, stream a_j synchronizes to stream a_m *viceversa*. Whenever either stream is ahead of the other, it waits and "*meets*" the other. In a 1-way dependency constraint $d_{jm}(a_j \rightarrow a_m)$, stream a_m synchronizes to stream a_j. Whenever a_m fails to "*meet*" (is ahead of) a_j, a_m waits for a_j. Finally, in a 0-way dependency constraint, neither stream synchronizes to the other. That is, streams a_j and a_m are unrelated; neither one waits. A constraint/update propagation matrix is used to enforce the multiple dependency constraints. This framework is described in detail [7].

In designing the framework we reused some ideas from the literature. To preserve temporal correspondence on the playback of asynchronous media, we relied on relative synchronization. To amortize overheads, we relied on temporal intervals. To enforce temporal intervals under relative synchronization, we relied on point synchronization (sync-events). To enforce relationships between streams, we use a master-slave model. To provide a flexible architecture, we relied on a slave-initiated "*publish-and-subscribe*" model to point synchronization, which facilitates the relative synchronization across multiple streams. We focused on the subset of relationships that can be decomposed into multiple binary, point-based, relationships. To account for asynchronous presentation, we coupled the relative interval-based model with a timing departure model resulting in an asynchronous relative interval-based model.

3 Experience

We have some baselining experience with this framework and the notion of long-term assessment of process performance. First, we provide an example of the analysis of long-term process trends specific to the domain of multimedia networking. Second, we examine an application of online ARC in multimedia scheduling. Last, we propose an application of online ARC for handling the integration of heterogeneous media types.

Example 1: Long-Term Trend Monitor

To detect long-term trends, we need to look into the process history. The father we look, the grander the patterns we see. Putting this paper in perspective with other research, we infer that internet traffic is really an aggregation of short-term, medium-term, and long-term trends. The analysis window basically defines the scope of what are the feasible mechanisms to address such variability.

The online algorithm takes a time series t_i and computes the a running sample mean and variance with a moving window. Each sample in the moving window is given equal weight. By the Central Limit Theorem, the sample mean is a random variable with normal distribution, which allows us to build online statistical tests as well as to derive confidence levels.

To smooth short-term fluctuations, we use a moving window of size $m = 120$ as defined below.

$$FMW(y_i, m) = \left\{ \begin{array}{l} \mu(i,m) = \mu(y_{i=i..i+m}) \\ \sigma(i,m) = \sigma(y_{i=i..i+m}) \end{array} \right\} \quad (13)$$

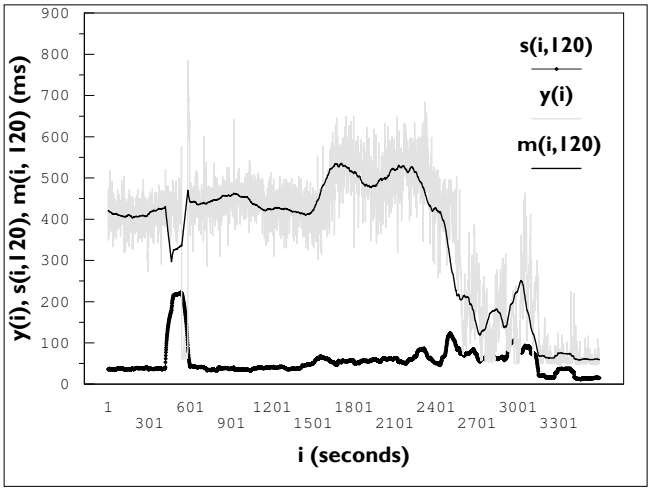

Figure 8. Long-term trends need to be accounted for. Shown above is a long-term moving window of 120 seconds observed over the same process y_i over a period of one hour. We formulate process controls that account for long-term variability in the streaming process.

Note that FWM is defined into the future. The performance confidence of the FWM over the time series ϵ_i (for $m = 120$) is shown in Fig. 9. It shows FWM is a very robust process indicator useful to understand the performance of long-term trend analysis.

This time, we rely on a backward time-series definition of the smoothing moving window for both $\sigma(i,m)$ and $\mu(i,m)$. Specifically,

$$BMW(y_i, m) = \left\{ \begin{array}{l} \mu(i,m) = \mu(y_{i=i-m..i}) \\ \sigma(i,m) = \sigma(y_{i=i-m..i}) \end{array} \right\} \tag{14}$$

To determine whether the current measurement belongs to a trend, we rely on hypothesis testing for determining whether the current $\sigma(i,m)$ is the same as a recently past $\sigma(j,m)$, where recently past means that $i - j < m$. Specifically we used $j = i - \frac{m}{2}$. Equivalently,

$$H_0 : \mu(i,m) = \mu(i - \frac{m}{2}, m) \tag{15}$$

To support online testing of the hypothesis, we rely on the statistic:

$$Z_0 = \mu(i,m) - \mu(i - \frac{m}{2}, m) \tag{16}$$

and reject H_0 if:

$$|Z_0| < r * \sigma(i,m) \tag{17}$$

where r is the number of σ-levels to tolerate (associated with confidence $P(|Z| > r) = \alpha$).[2] To understand this test, consider $r = 4$ (i.e., $P(|Z| > 4)$), the hypothesis tests whether the sample means for two overlapping windows sharing $\frac{m}{2}$ samples are the same by determining if the means of the non-overlapping segments are "close enough". This test is extremely conservative in considering two means to be close enough whenever their difference falls within 4σ range.

The performance of the online monitor for $m = 120$ and $r = 4$ is shown in Fig. 10. It shows the time series $y_i = roundtrip(x_i)$, together with the application of the backward-defined moving window, which results in a running $sigma(i,m)$ and $\mu(i,m)$. The online monitor transforms the incoming time series into piece-wise linear

[2]— Note that this is only a rough approximation test and that the proper test is a t-test on unknown mean and variance for a normal population with statistic $t_0 = \frac{\mu(i,m) - \mu(i-m/2,m)}{\sqrt{\frac{\sigma(i,m)^2 + \sigma(i-m/2,m)^2}{m}}}$, and test value of $t_{\alpha,m+1} \approx 3.3$.

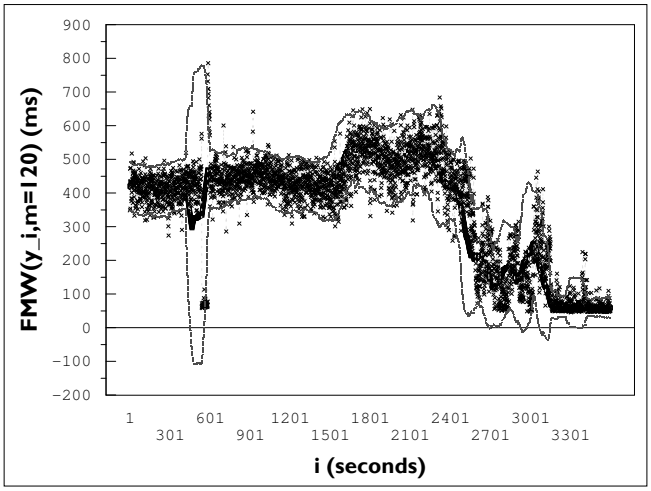

Figure 9. The 80% confidence region spanned by $[-2\sigma_i, 2\sigma_i]$ over the FWM() online monitor. Note the performance of the smoothed FWM indicators with respect to the behavior of original time series ϵ^i.

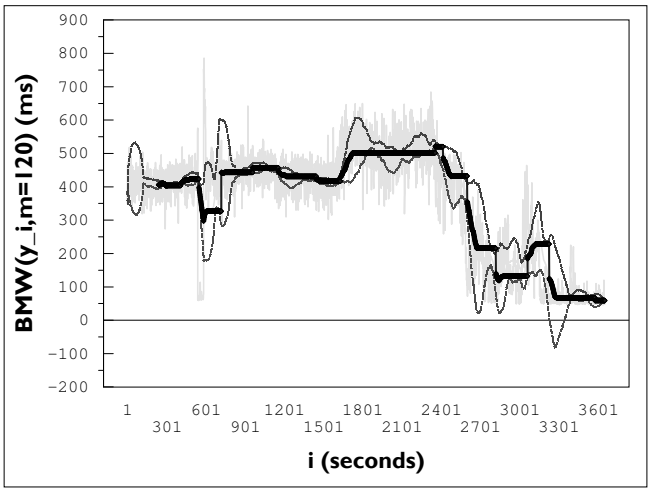

Figure 10. The behavior of the BWM online-monitor. The smoothed BWM indicators are coupled with an equality hypothesis test that compares a forecast of the mean with respect to the past. The result either changes the process mean or leaves it unchanged. The coupling of BWM with this hypothesis test (at a relax tolerance) smooths the ϵ_i^* into piece-wise linear segments that are either horizontal (no change) or with a slope (a shift in the process mean) under the presence of long-term trends.

segments of two kinds: horizontal segments and segments with a slope. Horizontal segments represent time intervals for which the process indicator (in this case, the network roundtrip delay) remains stable. Segments with a slope represent time intervals for which their is a process shift (in this case, there is a buildup or reduction of network delay). These segments provide a long-term approximation model to trends in network variability.

Statistical process control allows us to rely on relatively few samples to make strong inferences about the quality of a process and its behavior in terms of its mean and its standard deviation. Its robustness is derived from that it places no assumptions about the distribution of the random variable being sampled (since it analyzes their means which are normally distributed). SQC results in low sampling overheads that yield predictable confidence levels. In our internet and multimedia networking domains, these properties are very attractive. For example, this example could be used to monitor long-term trends at well-known spatially-stable clusters with relatively low overheads and yet high confidence. These long-term trend monitors and confidence intervals might prove useful to routing and load balancing nodes overseeing network traffic between these clusters.

Example 2: Adaptive Scheduling

The setup of this example is as follows. We have two streams, an audio and a window stream (A and W, respectively), under relative streaming on a workstation. The window stream is to adapt to the audio stream. Its degree of freedom $df(W)$ was defined to be elastic schedule time as in [6, 8]. We defined $y_i = g(x_i)$ to be synchronous presentation of audio-annotated window events, where x_i is a scheduling interval composed of one audio frame and the associated window events to that interval. We measured $\epsilon_i(A, W)$ every second and smoothed it with a window of two seconds into $\epsilon_i^*(A, W)$. Then, we formulated the following SQC problem:

$$SQC(\epsilon_i^*(A, W) \rightarrow 0, (T(A, W) - \sigma_i^*(A, W)) \rightarrow 0). \tag{18}$$

We coupled statistically significant departures, over on the smoothed asynchrony, of 1σ and 2σ with a graded compensation effort of $\{c, 10c\}$, respectively. The compensation constant was chosen to be $15ms$ (the minimum clock grain). The tolerance of the $T(A, W)$ was chosen to be $250ms$ so that the 2σ level was $500ms$ (i.e., 80% of the time, the presentation would have been in the range of audio-annotated foils — $[250 - 500]ms$). Because the feedback loop was ran within the workstation, the feedback latency was in the order of $[50]ms$ when accounting for thread switching and similar overheads. To compensate for such relatively large feedback delay we sensitize the control chart with 2σ limits. Periodicity lesser than $2s$ was found not suitable for continuous playback because of its high overhead.

The P3 policy relies on long-term scheduling corrections. Synchronization is 1-way; whenever b is ahead of a, b waits for a. The long-term scheduling compensation is computed during run-time, based on the statistical process performance of the relationship (see Fig. 11). Our scheduling controls implement long-term measures over the scheduling of a stream as follows. For a given relationship, we specify a tolerance region $(\mu_\epsilon, \sigma_\epsilon)$ over a smoothed indicator of the inter-stream asynchrony ϵ_i. The tolerance region is used to characterize the compensation effort with respect to the asynchrony departure. Fig. 12 shows the adaptive performance of this protocol in removing long-term asynchrony trends during playback. The strength of the compensation effort is related to the statistical significance of the smoothed asynchrony departure.

Example 3: Adaptive Synchronization

Long-term synchronization to continuity tradeoffs are specifiable. We show in this example, the use of synchronization effort as a degree of freedom resource (when coupled with SPC). The variability on the presentation of a media frame x_i has two components: (1) network variability and (2) synchronization effort (i.e., the amount of synchronization inserted in the presentation of frame x_i). The synchronization effort is dependent on the asynchrony ϵ_i^*. First, by carefully allocating synchronization effort for treatment of ϵ_i, we can control control its impact over the variability of the overall presentation process. That is, within our framework, we want to trigger synchronization effort only when the asynchrony ϵ_i^* is statistically significant. Then, by relying on long-term process controls, we can stabilize monitor the effectiveness of the allocation of synchronization effort with respect to the frequency of statistically significant asynchrony departures ϵ_i^*.

There are three possible asynchrony conditions (shown in Fig. 13) between two streams a and b: (1) a is ahead of its b (R_L); (2) a is behind of its b (R_R); and (3) a is in-sync to its b (R_C). The central region

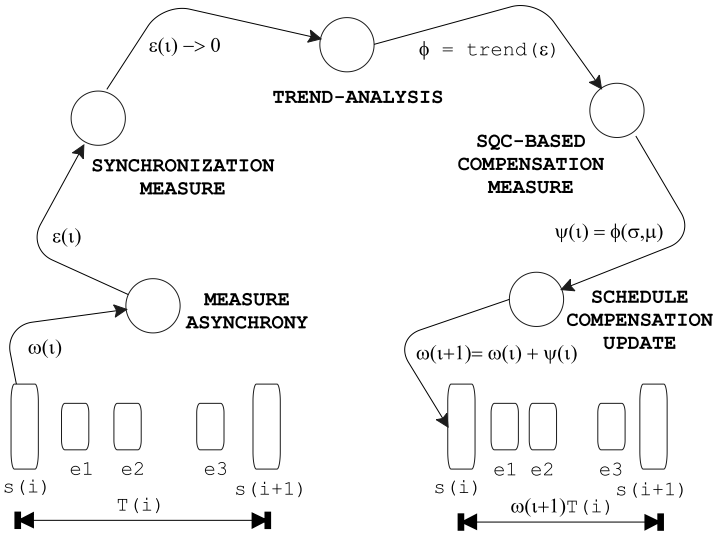

Figure 11. Process control model for long-term adaptive scheduling. A periodical synchronization event triggers a measurement of the current asynchrony ϵ_i. Synchronization is applied to reset the asynchrony. The asynchrony is smoothed into ϵ_i^* and statistical quality controls are applied over the smoothed asynchrony. The compensation effort $f_i = \omega_i$ is computed based on the statistical significance of ϵ_i, which is then used to scale (up/down) the logical time system of the stream.

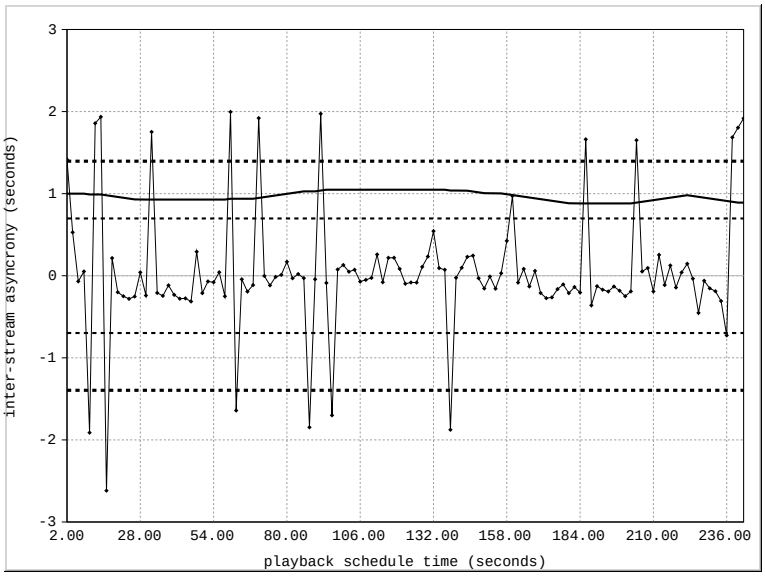

Figure 12. `SQC/SPC` **plot for P3 under higher load conditions. The compensation effort ω_i (the solid line fluctuating around 1) adapts to long-term trends in the asynchrony ϵ_i^* (the line of connected diamonds with random fluctuations around 0). Note that ω_i detects and matches each of the three trends on ϵ_i^*.**

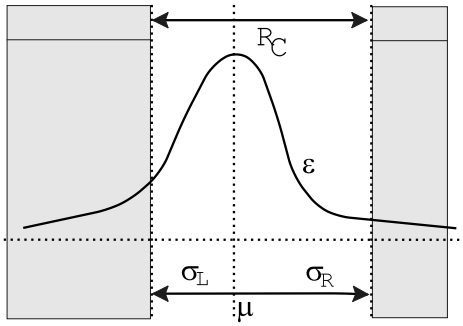

Figure 13. The region R_C represents an asynchrony tolerance region for which a and b are considered "relatively" synchronized. In this region, ϵ_i is regarded as non-significant. Synchronization measures are fired only when ϵ_i falls outside R_C. By varying the specification $T_{R_k} = (\sigma_L, \sigma_R)$ of this region, we can control the synchronization effort as just another degree of freedom resource in our framework.

R_C represents an asynchrony tolerance region for which a and b are considered "relatively" synchronized. In this region, ϵ_i is regarded as non-significant. Synchronization is triggered only when ϵ_i exceeds this tolerance range. By varying the specification (σ_L, σ_R) of this region, we can control the synchronization effort imposed over a stream. To further improve playback continuity and smoothness, we can couple this mechanism with adaptive scheduling.

The P5 policy relies on long-term scheduling corrections. Synchronization is relaxed 2-way. Whenever b is ahead of a, b may wait for a and whenever a is ahead of b, a may wait for b. This relaxation of synchronization is based on the run-time statistical process performance of the relationship. The relaxation of synchronization firing controls is implemented as follows (see Fig. 14). For a given relationship, we specify a tolerance region $(\mu_\epsilon, \sigma_\epsilon)$ over a smoothed indicator of the inter-stream asynchrony ϵ_i. The tolerance region is used to characterize the relaxation effort with respect to the statistical significance of the asynchrony departure. Blocking synchronization measures are initiated when a statistically significant trend and/or departure is detected. Fig. 15 illustrates the intended relaxation of synchronization by P5 over the behavior of the asynchrony and the associated synchronization-based discontinuities. The net effect of P5 is that (1) *asynchrony floats* between `LSL` and `USL`) control limits while (2) playback discontinuities due to synchronization occur only when the asynchrony exceeds such limits. Consequently, tradeoffs between asynchrony and continuity are now specifiable: (1) the asynchrony ϵ_i is controlled by the tolerance region ($|USL - LSL|$) and (2) discontinuities are now characterized by the probability of the asynchrony exceeding such tolerance $2P(|\epsilon| > USL)$.

4 Conclusion and Future Work

We showed the modeling and application of statistical process control (SPC) to adaptive streaming of multimedia. Specifically, we showed the application of SPC as a way to manage the end-to-end performance of the delivery of multimedia within a formal statistical framework. To facilitate the integration of application-oriented requirements, our measurements interface between client and server were based on statistical process performance measurements (e.g., 3σ departures) over application-oriented measurements as opposed to media-oriented measurements (3% packet loss). The proposed framework extends to various application scenarios with different synchronization and presentation requirements.

For true performance evaluation and control, application-oriented performance and network delivery performance should not be mutually exclusive but rather part of a common measurements system. Consider the following example. Suppose that a given streaming process uses 10 packets to deliver a video frame and suppose that the streaming process is loosing about 1 out of every 10 packets. Under the network-oriented model, we could measure the performance of the streaming process as the packet loss ratio (90%). However, under an application-oriented model, we would measure such performance in terms of the number of frames received. With compressed video the loss of any portion of a frame may make the entire frame unusable and

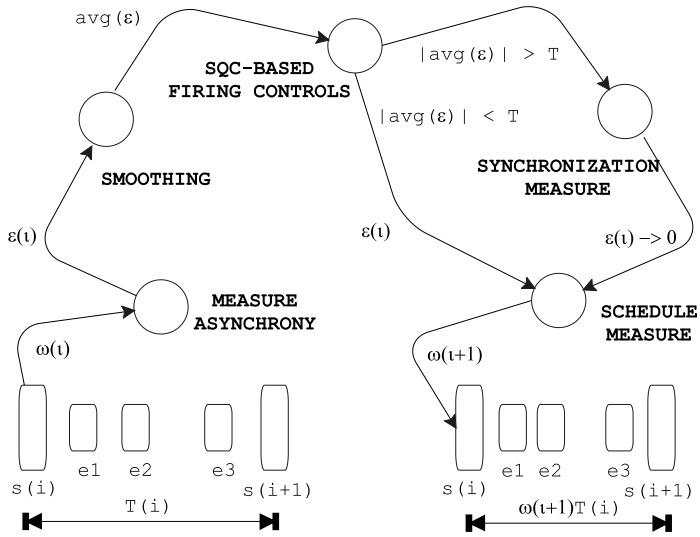

Figure 14. Process control model for long-term adaptive synchronization controls. A periodical synchronization event triggers a measurement of the current asynchrony ϵ_i. The asynchrony is smoothed into ϵ_i^* and statistical quality controls are applied over the smoothed asynchrony to determine whether ϵ_i is statistically-significant with respect to process history. In such cases, synchronization is applied to reset the asynchrony. Otherwise, the asynchrony ϵ_i^* is deemed tolerable and no synchronization is applied.

consequently the 10% packet loss really represents a 100% frame loss. Clearly, the process needs to be adjusted since the use of SPC alone results in adjustments only if the resulting departure is statistically-significance. Thus, finer-grain controls are needed.

By combining our findings, so far, on long-term network variability, with results from a large number of short-term probe-and-adapt protocols, and with recent findings on very-long-term network stability [2], we could infer that network variability is really composed of overlapping (1) short-term (measured in milliseconds), (2) long-term piece-wise trends (measured in minutes), and (3) very-long-term (measured hours) trends. It appears then that to truly address network variability, an integrated solution is needed to target each component as part of a whole. Process controls should be used to guide short-term adaptive rate controls in a consistent direction, and with complementary compensation, to that of long-term adaptive rate controls.

References

[1] L. Aroian and H. Levine. The effectiveness of quality control charts. *Journal of the American Statistical Association*, 44, 1950.

[2] H. Balakrishnan, M. Stemm, S. Seshan, and R. Katz. Analyzin stability in wide area network performance. In *ACM Sigmetrics Conference on Measurements and Modeling*, June 1997.

[3] Western Electric Corp. *Statistical Quality Control Handbook.*

[4] J. Hunter. The exponentially weighted moving average. *Journal of Quality Technology*, 17, 1986.

[5] K. Jeffay, D. Stone, and F. Smith. Transport and display mechanisms for multimedia conferencing across packet switched networks. *Computer Networks and ISDN Systems*, 26(10):1281–1304, July 1994.

[6] M.Y. Kim and J. Song. Multimedia documents with elastic time. In *Proc. of ACM Multimedia '95*, pages 143–154, San Francisco, CA, November 1995.

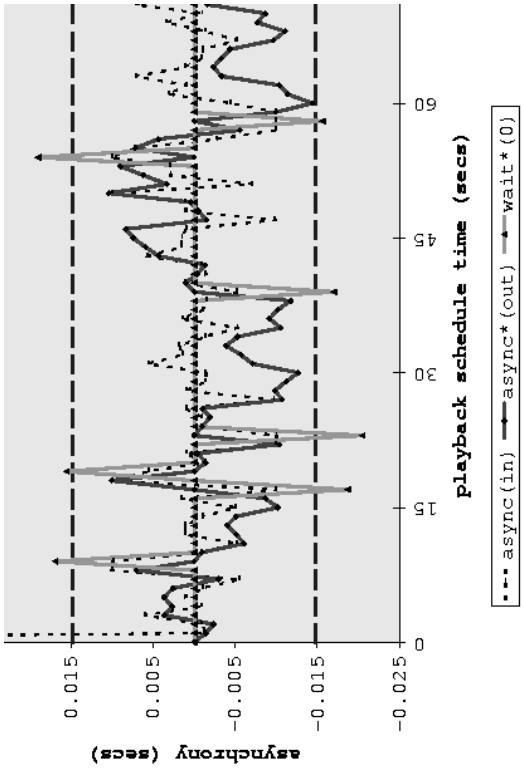

Figure 15. (Please note that this figure is rotated.) A (simulated) illustration of the effects of the relaxation of synchronization effort $wait_i$. The relaxation is achieved by expanding the tolerance region over the smoothed asynchrony $\epsilon_i^* = async^*(out)$ for of a stream in a binary relationship. The original $\epsilon_i = async(in)$ is shown for comparison.

[7] N.R. Manohar. Interactive time delayed sharing of computer supported workspaces via the streaming of re-executable content. Technical Report Ph.D. Thesis, Department of Eletrical Engineering and Computer Science, University of Michigan, 1997.

[8] N.R. Manohar and A. Prakash. Dealing with synchronization and timing variability in the playback of interactive session recordings. In *Proc. of ACM Multimedia '95*, pages 45–56, San Francisco, CA, November 1995.

[9] D. C. Montgomery. *Cumulative-Sum and Exponentially Weighted Moving Average Control Charts*, pages 279–310. John Wiley and Sons, 1990.

[10] D. C. Montgomery. *Statistical Process Control*, pages 99–144. John Wiley and Sons, 1990.

[11] S. Ramanathan and Venkat P. Rangan. Adaptive feedback techniques for synchronized multimedia retrieval over integrated networks. *ieanep*, 1(2):246–260, April 1993.

[12] D.L. Stone and K. Jeffay. An empirical study of delay jitter management policies. *ACM Multimedia Systems*, 2(6):267–279, January 1995.